

# RasPi

DESIGN  
BUILD  
CODE

5

Get hands-on with your Raspberry Pi

63  
PAGES  
OF PI



## Hack a

# BIGTRAK

Plus Animate Pivaders



# Welcome



Incredibly rewarding though it is to create your own Raspberry Pi project from scratch using only the components you've got on your workbench, there's just something so satisfying about finding a cool electronic device and then tearing it apart to give it some Pi brains. That's what we're doing this issue with the Bigtrak – a brilliant RC vehicle that's seen a renaissance in recent years and can easily be hacked with your Pi. And we're also finishing up the Pivaders project that we started last issue, adding in more animation to show you how to use sprite sheets and even throwing in some sound effects. Plus we've got a few neat competitions for you over in Talking Pi. Happy hacking, everyone!

*Gavin Thomas*

Deputy Editor

From the makers of  
**LinuxUser**  
& Developer

Join the conversation at...

 @linuxusermag

 Linux User & Developer

 RasPi@imagine-publishing.co.uk

## Get inspired

Discover the RasPi community's best projects

## Expert advice

Got a question? Get in touch and we'll give you a hand

## Easy-to-follow guides

Learn to make and code gadgets with Raspberry Pi







# Contents

---

## Hack a Bigtrak

Give it a Pi brain and drive it with a PS3 controller



## Supercharge your Raspberry Pi

Portable touchscreen

Barometric pressure sensor

Electret microphone

Motor controller

Stereo amplifier



## What is the VideCore graphics stack?

...and now that it's open, what happens next?



## Add animation and sound to Pivaders

Complete the final level of your arcade game



## XLoBorg tilt controller

Play Pivaders by picking up your entire Pi



## Aber Sailbot

Autonomous robot yacht that competes worldwide



## Talking Pi

Your questions answered and your opinions shared





A white Sony Bigtrak toy car is shown from a three-quarter perspective. The car has a transparent top section that reveals internal electronic components, including a Raspberry Pi board, a game controller, and various cables. The car is decorated with a red and yellow stripe along its side, and the word "bigtrak" is printed in black. A red line points to the side of the car, and a grey line points to the game controller. The car is positioned on a white surface.





The Raspberry Pi is a small, low-cost computer designed to promote an interest in computing and programming – but it doesn't have to be straight-laced computing. In fact, in this article we'll be showing you how you can use it to turn a Bigtrak into a robot. That's educational, right?

The Bigtrak is a toy that takes in a list of straightforward commands (like go forwards, turn left, turn right) and then executes them. To make things more interesting we're going to remove the existing circuitry and replace it with a Raspberry Pi, using a small motor driver to safely control the motors in the Bigtrak, which we'll then set up to be controlled via a PlayStation 3 DualShock controller.

Everything required on the software side comes pre-installed on the latest Raspbian OS images, so all we need to translate changes from the controller to the motors is a small Python script that uses the Pygame and RPI.GPIO modules.



## THE PROJECT ESSENTIALS

### Bigtrak

[bigtrakxtr.co.uk](http://bigtrakxtr.co.uk)

### Motor driver

[bit.ly/1iOnFug](http://bit.ly/1iOnFug)

### USB Battery pack

[amzn.to/1h2PBil](http://amzn.to/1h2PBil)

### Breadboard

### PS3 DualShock controller

## 01 Opening up the Bigtrak – the easy bit

Before we can make any changes to the Bigtrak we need to get inside. First, flip the Bigtrak upside down and remove the nine screws from around the edge. These are mostly easy to get at, however the ones on the front may require a more slender screwdriver to reach them.

## 02 Opening up the Bigtrak – the fiddly bit

The last two screws are located underneath the grey grille on the back. This grille is held in place by four plastic tabs that need to be pushed in while at the same time sliding the grille away from the Bigtrak. This can be

“All we need to translate changes from the controller to the motors is a small Python script that uses the Pygame and RPI.GPIO modules”





quite tricky as there is limited space to get extra hands in to help out. It can help to wedge some thin plastic items (like a guitar pick) into the sides to keep those two tabs unlocked, while using your fingers to push in the bottom two tabs and slide the grille upwards, allowing you to remove the screws.

### 03 Removing the top

Put the Bigtrak back onto its wheels then carefully loosen the top and lift upwards. The lid is connected to the base with a ribbon cable and a switch, so only pull the top up far enough for you to tilt it to one side and expose the inside.

With the lid lifted up onto one edge, remove the screw holding the switch in place and detach it from the lid. Next, you need to unscrew the two screws on the PCB that hold the ribbon cable in place and let it slip free.

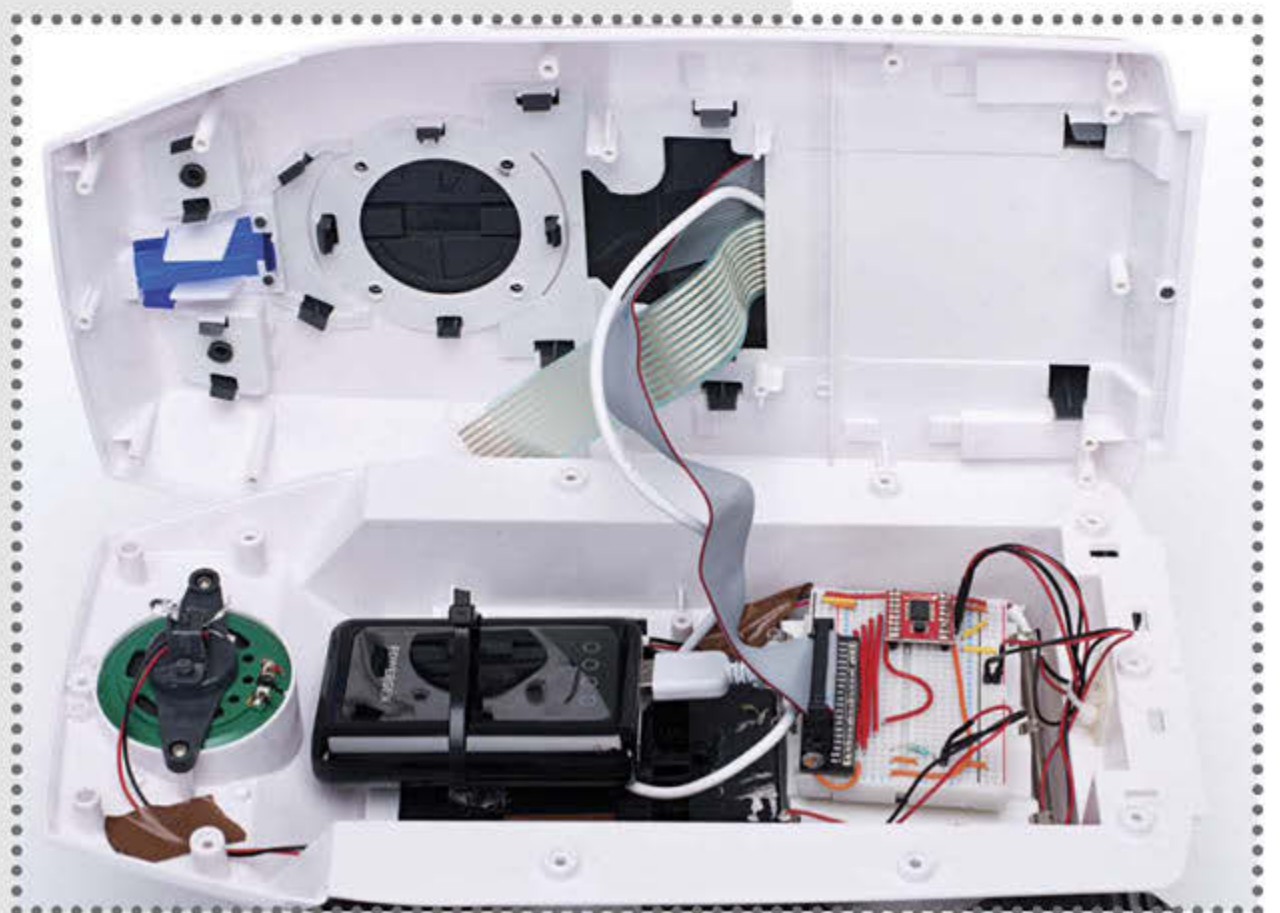
With the switch and ribbon cable disconnected, the lid should now come free and can finally be completely removed from the base of the Bigtrak.

“Cut the wires leading to the main PCB. The ones for the switch and power should be cut close to the PCB”

**Below** Be gentle when removing the lid so you don't yank the ribbon out by mistake

### 04 Cut the wires

Cut the wires leading to the main PCB. The ones for the switch and power should be cut close to the PCB (so we can reuse them later) whereas the ones to the LED and speaker can be cut wherever you like.



## 05 Remove the engine

Turn the Bigtrak upside down and remove the four screws holding the engine in place (this will reduce the chance of soldering iron damage to the main body). Carefully turn the Bigtrak back over and lift it up until the engine slips free.

## 06 Rewire the motor

Remove the solder connecting the PCB to the motors (a solder mop is useful here) and then remove the PCB. With the PCB removed we can now attach wires to the motors in order to drive them from the Raspberry Pi, as opposed to the on-body commands. The wires will need to be long enough to reach the back of the Bigtrak, so be generous – after all, it's far easier to trim long wires to length than replace short wires entirely!

Having installed all of the wires, you can now replace the engine back into the Bigtrak.

## 07 Connect the motor driver

With the motors back in place we now need to build up a circuit to drive it from the Raspberry Pi. We've used a ribbon cable to connect the GPIO pins on the Raspberry Pi to a breadboard, before connecting it up to a Dual Motor Driver (<http://proto-pic.co.uk/motor-driver-1a-dual-tb6612fng>) to actually drive the motors. This keeps the higher voltage the motors require away from the sensitive GPIO pins.

The connections made on the breadboard are listed in the table on the next page. These values will be needed when writing the software and may be different depending on the breakout board you are using, and the Raspberry Pi revision.

“With the PCB removed we can now attach wires to the motors in order to drive them from the Raspberry Pi”

RPi GIPO	Motor Driver
24	AIN2
17	AIN1
18	STBY
21	BIN1
22	BIN2

“The breadboard is going to be installed on top of the battery compartment inside the Bigtrak”

With the PWMA and PWMB pins directly connected to the 3.3V power rail, the motors will now always run at full speed for as long as they're active.

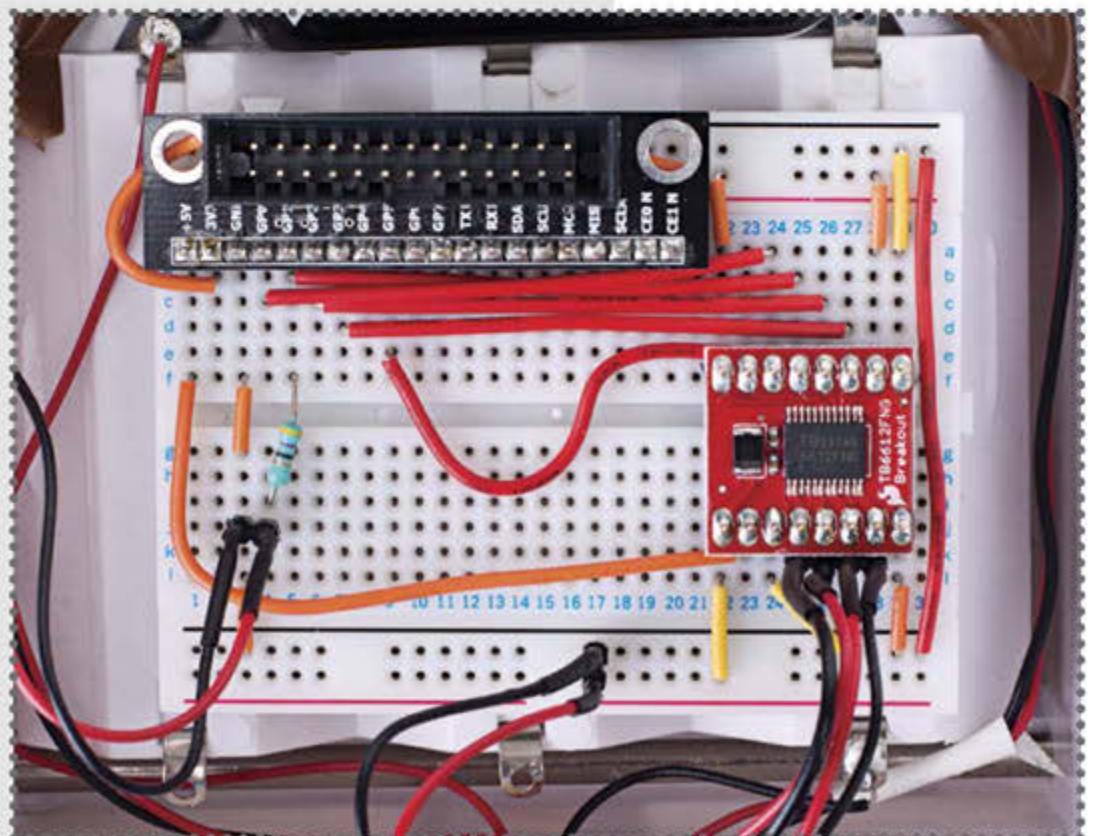
## 08 Install the breadboard

The breadboard is going to be installed on top of the battery compartment inside the Bigtrak, so the wires from the motors should be brought to the back of the unit and cable-tied into place. The wires to the batteries can also be brought back to the same place to help keep things tidy.

**Below** Follow the Fritzing diagram on the next page for a more detailed guide

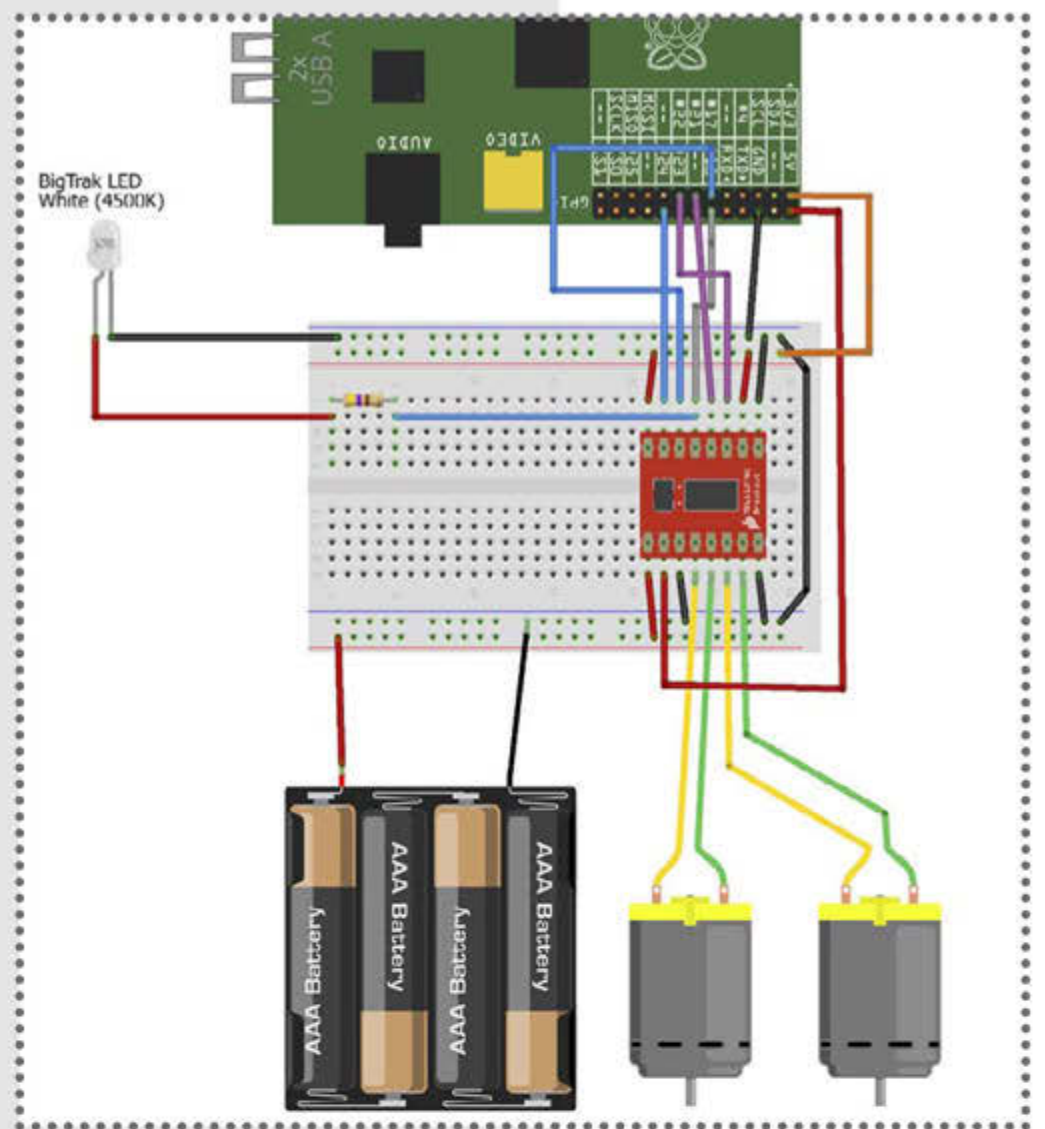
## 09 Wire it all together

In order to easily connect the motors and batteries to the breadboard we have soldered some modular connector plugs to the ends of the cable, allowing them to just push into place (these are available from [www.maplin.co.uk/modular-connectors-5348](http://www.maplin.co.uk/modular-connectors-5348)). With





the breadboard installed (sticking it into place for support) we can now, after double-checking all the connections, plug the motors and power into it. To know when the motors are enabled (and reduce the chance of unexpected movement), the LED can be connected to the breadboard so that it lights up whenever the 'standby' pin is active, using a resistor to ensure it doesn't pull too much current and go 'pop'.



**Above** We're using a Model B here but the B+ and A+ will be fine

## 10 Provide power

Power for the Raspberry Pi is supplied via a USB battery pack that is installed on top of the engine and can be held in place by a couple of cable ties or a Velcro strip. This type of battery is typically sold as a portable mobile phone or iPad charger – the one used here is rated at 8000mAh, able to power the Raspberry Pi for up to eight hours.

## 11 Connect to the Raspberry Pi – adding cables

As the Raspberry Pi will be mounted on the top of the Bigtrak, we need to run the ribbon and power cable through the top of the case. To do this, turn the top of the Bigtrak upside down and release the front two catches that hold the dark grey plastic in place – this provides a big enough gap to feed the ribbon cable and USB power cable through. Make sure that the

red edge of the ribbon cable correctly matches up with the connector on the breadboard in order to save yourself from having to twist the cable inside the case.

## 12 Connect to the Raspberry Pi – final steps

With the top of the Bigtrak back on, the Raspberry Pi can now be put in place, keeping the GPIO pins towards the front to allow the ribbon cable to easily connect. As for the battery pack, we're holding it in place with cable ties and sticky pads. In theory it's possible to attach the bare Raspberry Pi to the Bigtrak, however this can cause the SD card to press against the edge and bend, so it's recommended to use a case to protect the Raspberry Pi.

Connect the ribbon and power cable to the Raspberry Pi, turn it on and it's now ready to be told what to do. For setting up the software it may be easier to connect up a keyboard and monitor to the Raspberry Pi at this point.

## 13 Connect the PS3 controller

This should be a simple case of plugging the PS3 controller into one of the USB ports, as all the software to support it is included in the default Raspbian OS image and it will be automatically detected as a joystick. To confirm that the PS3 controller has been detected, run `lsusb` and checked that it appears in the resulting list.



**Above** You can get some great cases to fit your model of Pi from Adafruit

“All the software to support the PS3 controller is included in the default Raspbian OS”



## 14 Run the software

Now with the system all set up, it should just be a case of copying the 'bigtrak.py' file we uploaded to **bit.ly/1tMrYdC** onto your Raspberry Pi and running it. As the script accesses the GPIO pins, it will need to be run as the superuser, so launch it using:


```
sudo python bigtrak.py
```

Now we can control the Bigtrak using the analogue sticks! Moving the left stick will control the left motor and moving the right stick will control the right. So, to move forwards push both sticks up, pull both down to go backwards and push one up and one down to rotate on the spot.

If the analogue sticks are not controlling the Bigtrak as expected, just double-check the GPIO connections to make sure that they are all as detailed in the Fritzing diagram.

## 15 Next steps

Now that you have a solid base for your Raspberry Pi robot, you can make further adjustments to it. Possible next steps could be: add a USB Bluetooth adaptor so the PS3 controller can be connected wirelessly; replace the breadboard with a PiPlate or 'Slice of Pi' add-on board, allowing the Raspberry Pi to be installed inside the Bigtrak; connect up the Raspberry Pi camera and a USB Wi-Fi adaptor to stream video as you drive around; or add a robot arm!



“Connect up the Raspberry Pi camera and a USB Wi-Fi adaptor to stream video as you drive around”

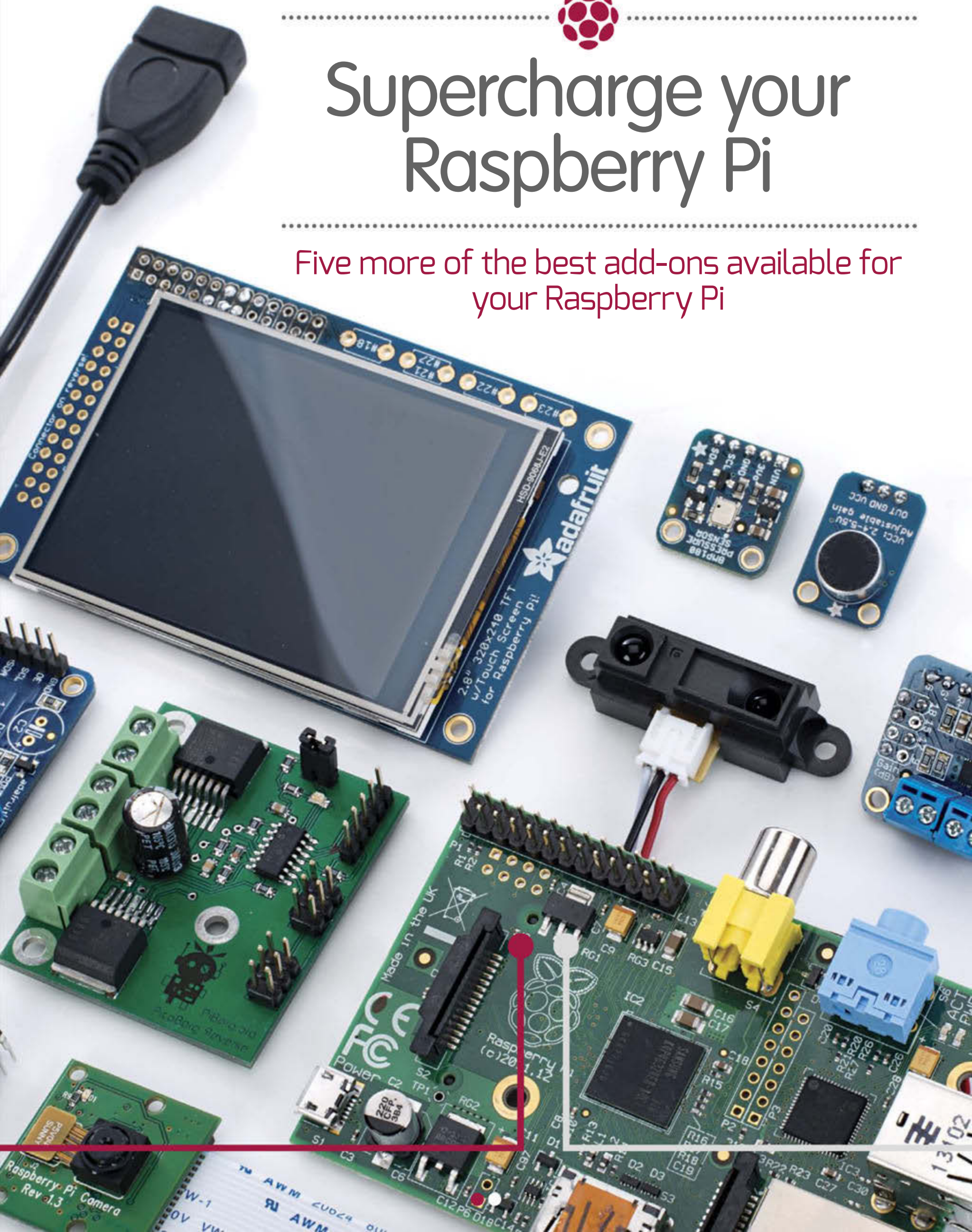




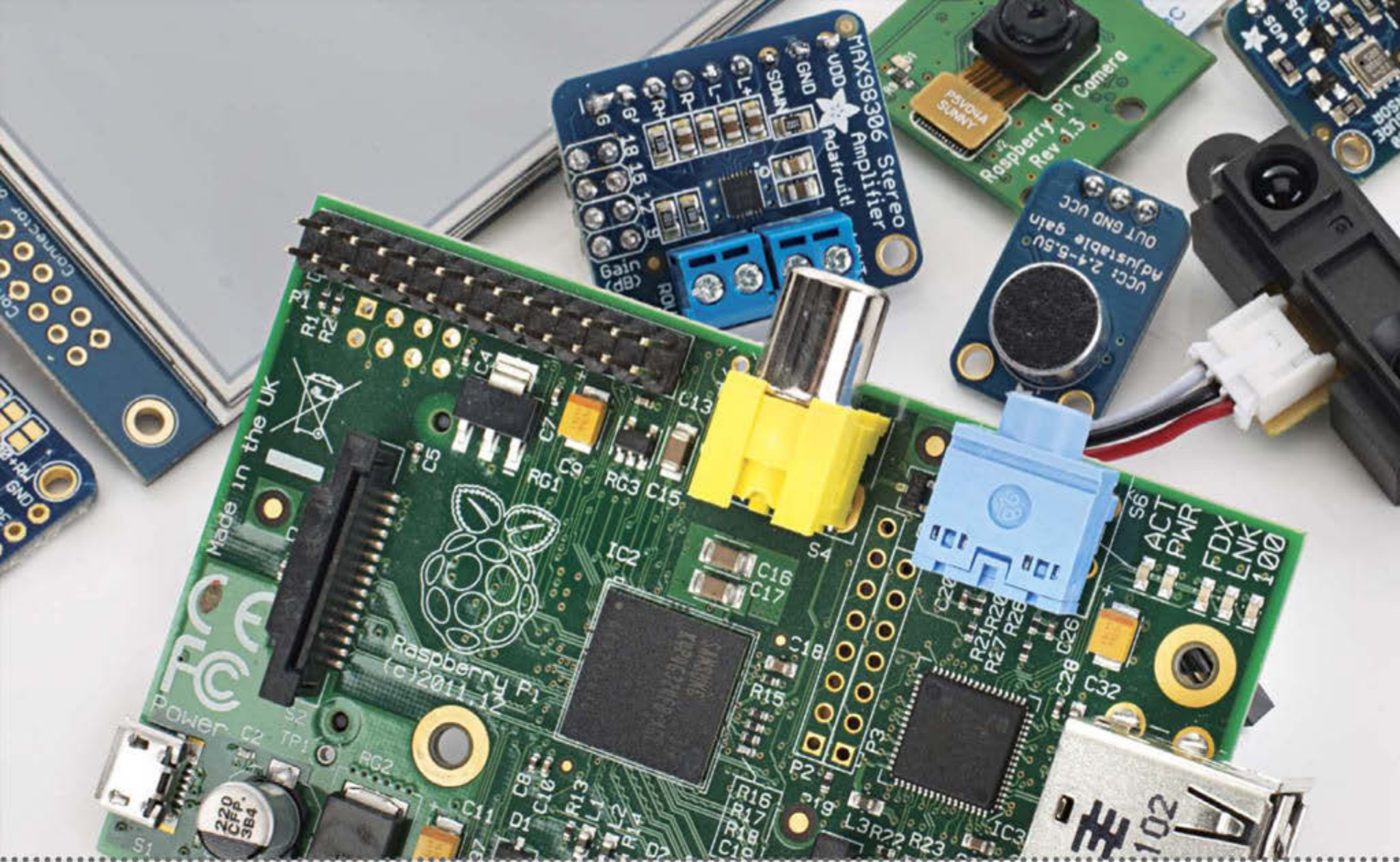


# Supercharge your Raspberry Pi

Five more of the best add-ons available for your Raspberry Pi







It's clear that the Raspberry Pi has been a revolution for small form factor computing, but it's not quite perfect – only so much technology can be packed into a tiny space, after all. For enthusiasts and makers there's a growing list of extra-curricular add-ons that can help make their projects, gadgets and Internet Of Things devices come to life.

In the previous issue of **RasPi** we took a look at five of our favourite add-on modules for the Pi, all of which would make a fantastic addition to your kit and many of which we regularly use in our own projects: analog to digital converters, infrared reflectance sensors, the Raspberry Pi camera module, USB power cables with switches and servo/PWM driver boards. To round off your collection we've got another five fantastic modules to recommend that will open up even more project possibilities.

“There's a growing list of add-ons that can help make projects, gadgets and Internet Of Things devices come to life”





# Portable touchscreen

Make your Raspberry Pi truly portable by attaching a touchscreen display



## What is it?

The PiTFT is a 2.8-inch capacitive TFT LCD touchscreen that's been specifically designed with the Raspberry Pi in mind by the project gurus over at Adafruit. It's capable of slotting directly on top of the Raspberry Pi and is about as big as the Pi is itself.

## Why do it?

There are numerous reasons why you'd want to add such a screen to a Raspberry Pi but they all generally come down to the fact that the Pi is very small and very portable and most monitors are not. While you could VPN in via a phone if

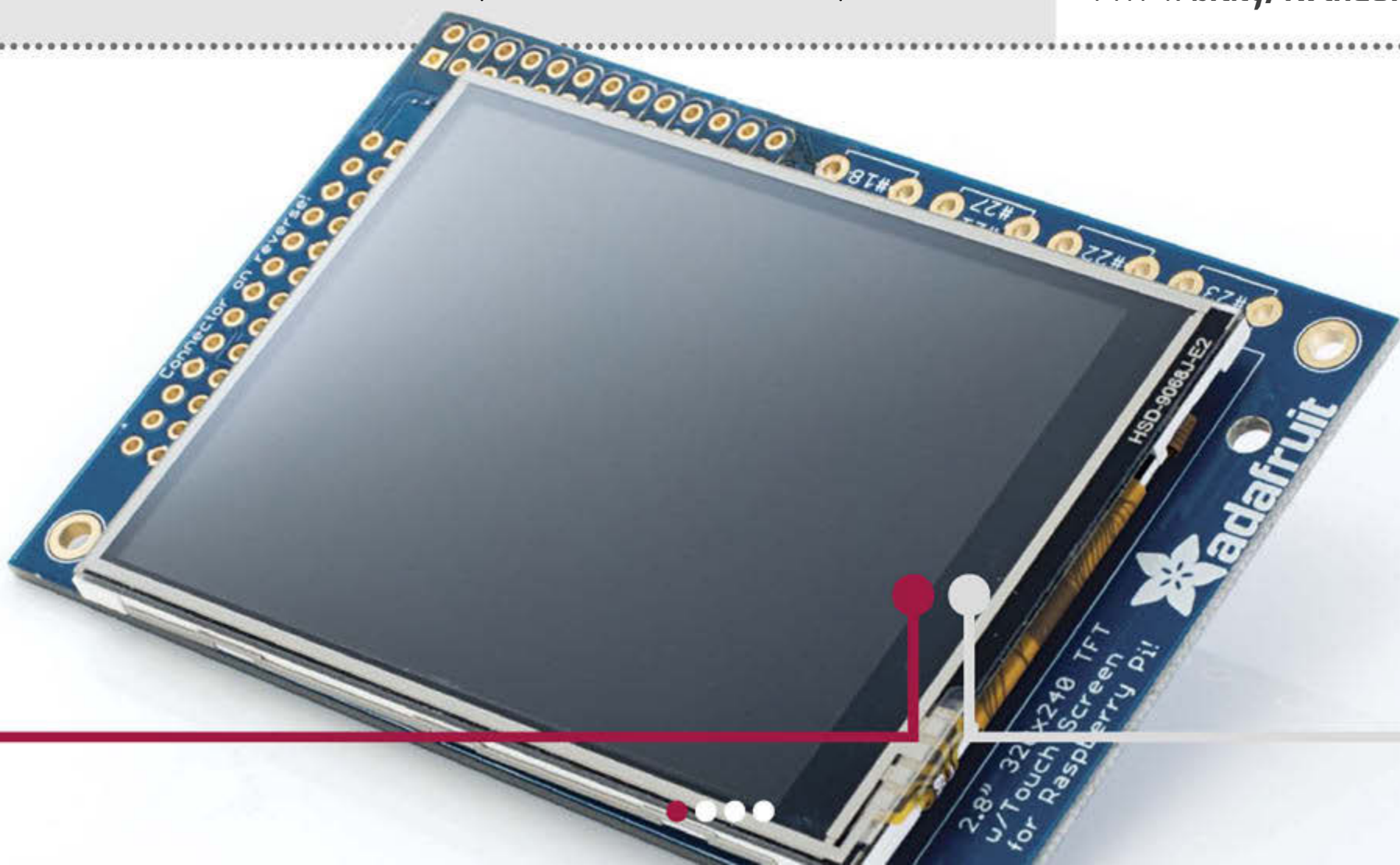


THE PROJECT  
ESSENTIALS

PiTFT Mini Kit  
- 320x240 2.8

TFT+Touchscreen  
[shop.pimoroni.com](http://shop.pimoroni.com)

**Below** Linux User & Developer has a guide to making a portable video player using PiTFT: [bit.ly/1wtnL0r](http://bit.ly/1wtnL0r)





you're on the go, the screen is connected directly to the Pi and doesn't involve awkward wireless networking. And since it's a touchscreen you don't need to bring along other input devices, as it's powered off the Raspberry Pi as well.

This opens it up to a world of possibilities. Portable computer, touchscreen control pad, video camera... anything that could benefit from your Raspberry Pi having a screen and human input while away from your main monitor.

## How to use it

All you need to get it to work is the PiTFT from our friends at Pimoroni – soldering is required. Turn the Raspberry Pi off and slot it into the GPIO. Make sure it's still connected to a conventional screen and turn it back on again, staying in command line mode. Right now the PiTFT will not turn on as you need to configure the Raspberry Pi, and more specifically Raspbian, to support it. Once you're logged in you can start the process by downloading files with:

```
$ wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-bin-adafruit.deb
$ wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-dev-adafruit.deb
$ wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-doc-adafruit.deb
$ wget http://adafruit-download.s3.amazonaws.com/libraspberrypi0-adafruit.deb
$ wget http://adafruit-download.s3.amazonaws.com/raspberrypi-bootloader-adafruit-112613.deb
```

Install them all with `sudo dpkg -i -B *.deb` and then give it a reboot. Once you've logged back in you can install the driver for the actual screen with:

“This opens it up to a world of possibilities. Portable computer, touchscreen control pad, video camera...”

```
$ sudo modprobe spi-bcm2708
$ sudo modprobe fbtft_device name=adafruitts
rotate=90
$ export FRAMEBUFFER=/dev/fb1
$ startx
```

At this point you should be getting a picture on the screen, however once you reboot it won't last. To exit it for now press CTRL+C. To make it work on boot we'll first open /etc/modules and add:

```
spi-bcm2708
fbtft_device
```

Save that file out and now open up the Adafruit configuration file with:

```
$ sudo nano /etc/modprobe.d/adafruit.conf
```

... and add the following line:

```
options fbtft_device name=adafruitts rotate=90
frequency=32000000
```

That's the screen rotated 90 degrees there, and now we need to do the same with the touch aspect to orient it properly. We need to create a directory and file to handle this:

```
$ sudo mkdir /etc/X11/xorg.conf.d
$ sudo nano /etc/X11/xorg.conf.d/99-
calibration.conf
```

In this new file you'll need to add:

“At this point you should be getting a picture on the screen, however once you reboot it won't last”



```
Section "InputClass"
    Identifier "calibration"
    MatchProduct "stmpe-ts"
    Option "Calibration" "3800 200 200 3800"
    Option "SwapAxes" "1"
EndSection
```

It's just about configured. Edit the '~/.profile' file by adding `export FRAMEBUFFER=/dev/fb1` and save. Now after a reboot every time you use `startx` you'll turn on the display; to have it turn on whenever the Pi is on install:

```
$ sudo apt-get install xserver-xorg-video-fbdev
```

After this, create the file `/usr/share/X11/xorg.conf.d/99-fbdev.conf` and add to it:

```
Section "Device"
    Identifier "myfb"
    Driver "fbdev"
    Option "fbdev" "/dev/fb1"
EndSection
```

Go to Raspbian configuration with `raspi-config`, enable the desktop and finally press Finish in order to perform a reboot. You should log in with the screen turned on. The screen is a little small and it's better for custom-designed interfaces, using Python or whatever you prefer. The touchscreen can also be calibrated with some extra software if it's not precise enough for you as well.

“The screen is a little small and it's better for custom-designed interfaces, using Python or whatever you prefer”



# Predict the weather

Use your Raspberry Pi to measure both air pressure and temperature



## What is it?

Adafruit's BMP180 is a barometric pressure sensor that allows you to detect changes in air pressure. Coupled with a temperature sensor it can predict short-term changes in the weather.

## Why do it?

Being able to predict the weather has the obvious benefit of knowing whether or not to bring an umbrella or sunglasses into work. However, you could also use it in home automation to open windows or close curtains when it's hot or about to start raining.

"Open windows or close curtains when it's hot or about to start raining"



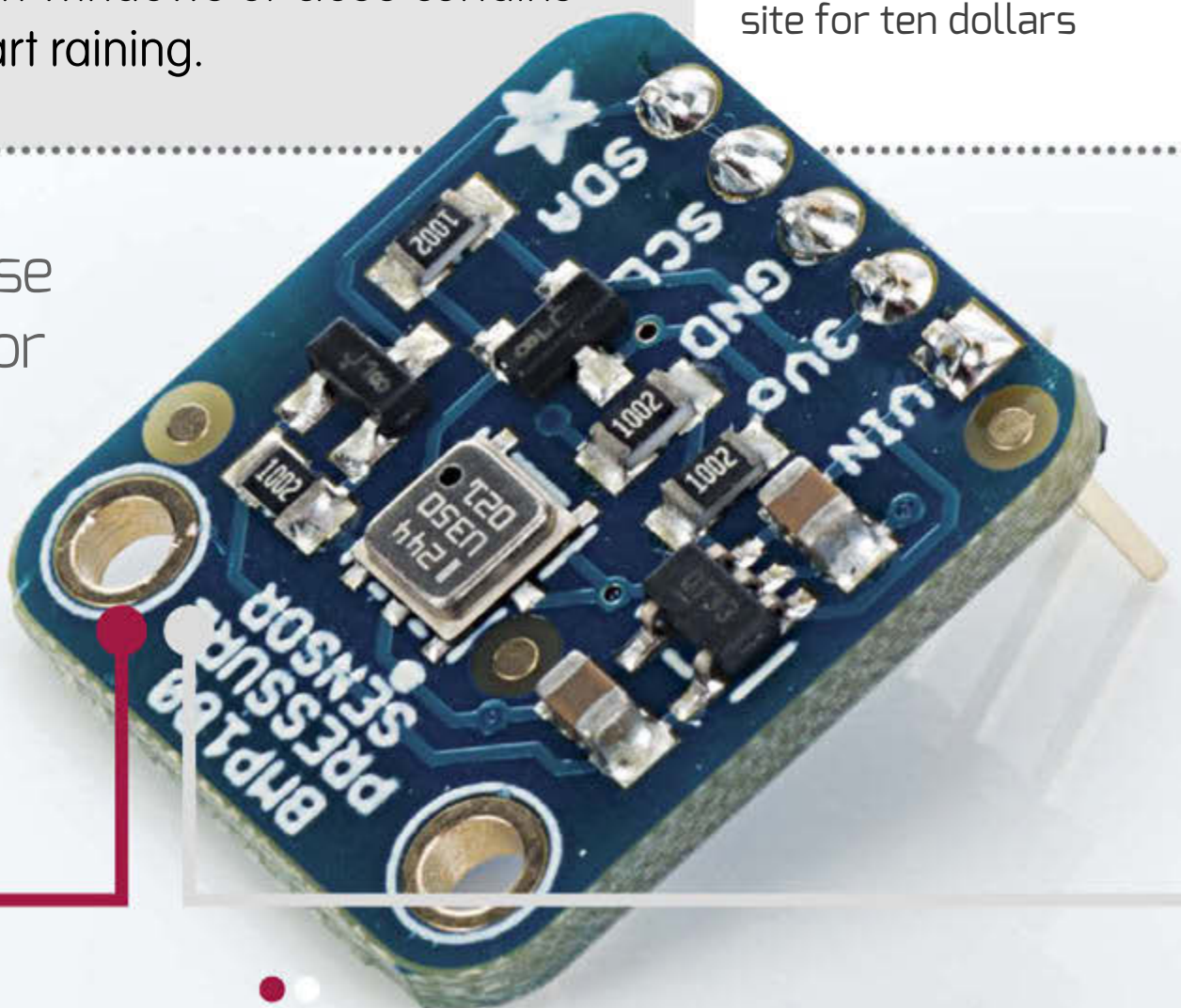
THE PROJECT  
ESSENTIALS

Adafruit BMP180

[modmypi.com](http://modmypi.com)

Breadboard  
Connectors

**Below** Adafruit sell the BMP180 on their site for ten dollars





# Get up and running

## 01 Enable I2C

In the terminal, open the modules file to activate the I2C pins by adding two lines.

After the following, reboot:

```
$ sudo nano /etc/modules  
i2c-bcm2708  
i2c-dev
```

## 02 Get the code

The software for the BMP is available from the Adafruit GitHub: [github.com/adafruit](https://github.com/adafruit). Download it and navigate to the relevant folder:

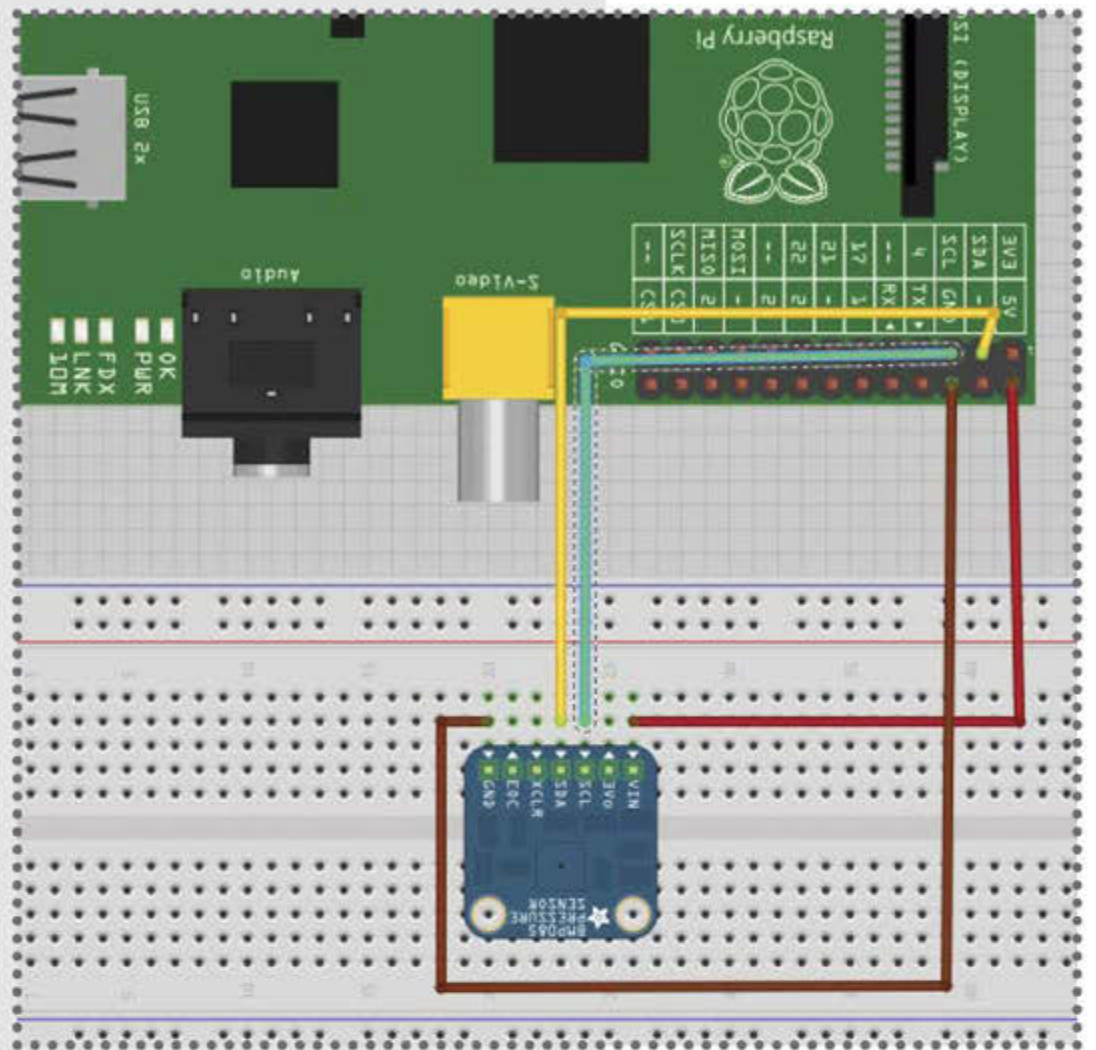
```
$ git clone https://github.com/adafruit/  
Adafruit-Raspberry-Pi-Python-Code.git  
$ cd Adafruit-Raspberry-Pi-Python-Code  
$ cd Adafruit_BMP085
```

## 03 Test your sensor

Now you're almost ready to get some data by first installing a package and then testing. You can do this in the terminal with the following commands:

```
$ sudo apt-get install python-smbus  
$ sudo python Adafruit_BMP085_example.py
```

“The software for the BMP is available from the Adafruit GitHub: [github.com/adafruit](https://github.com/adafruit)”



**Above** Soldering aside, breadboarding the BMP180 is simple



# Give your Pi ears

Measure sound for your next project affordably



## What is it?

An electret microphone is a clever piece of microphone design that uses a permanently charged material called electret, forgoing the need for a polarising power source. Electret effectively has a built-in static electric charge that won't decay for hundreds of years – some pretty clever stuff. The Adafruit Electret Microphone, shown here, comes with a built-in amplifier and is rigged and ready to use, bar soldering.

## How to use it

Simply connect it to your 3.3 voltage, ground and an analog input provided by an analog to digital converter.

You just plug output from the electret straight into a free ADC-enabled pin and you can measure the sound response from the microphone. If you want to get fancy, all you need to do is convert the values coming from the ADC to a PWM-compatible value (simply divide it by four) and you can use your voice to trigger the brightness of an LED light.



THE PROJECT  
ESSENTIALS

Adafruit Electret  
Microphone Amp  
[modmypi.com](http://modmypi.com)

**Below** Try a clap switch or an FM transmitter for your first project with this







# Control motors and steppers

Controlling motors on your Pi is child's play with PiBorg



## What is it?

There's a fundamental drive instilled in all tinkerers and it goes something very much like, 'If you can, add motors to it'. The Raspberry Pi is no exception and in fact, adding motors to the Pi has become a cottage industry of its own, with everyone and their Kickstarter-funded dog looking for a piece of the action. There are countless ways you can power motors with the Raspberry Pi, starting from a simple IC chip – the good old L293D – that can be breadboarded with relative ease. The

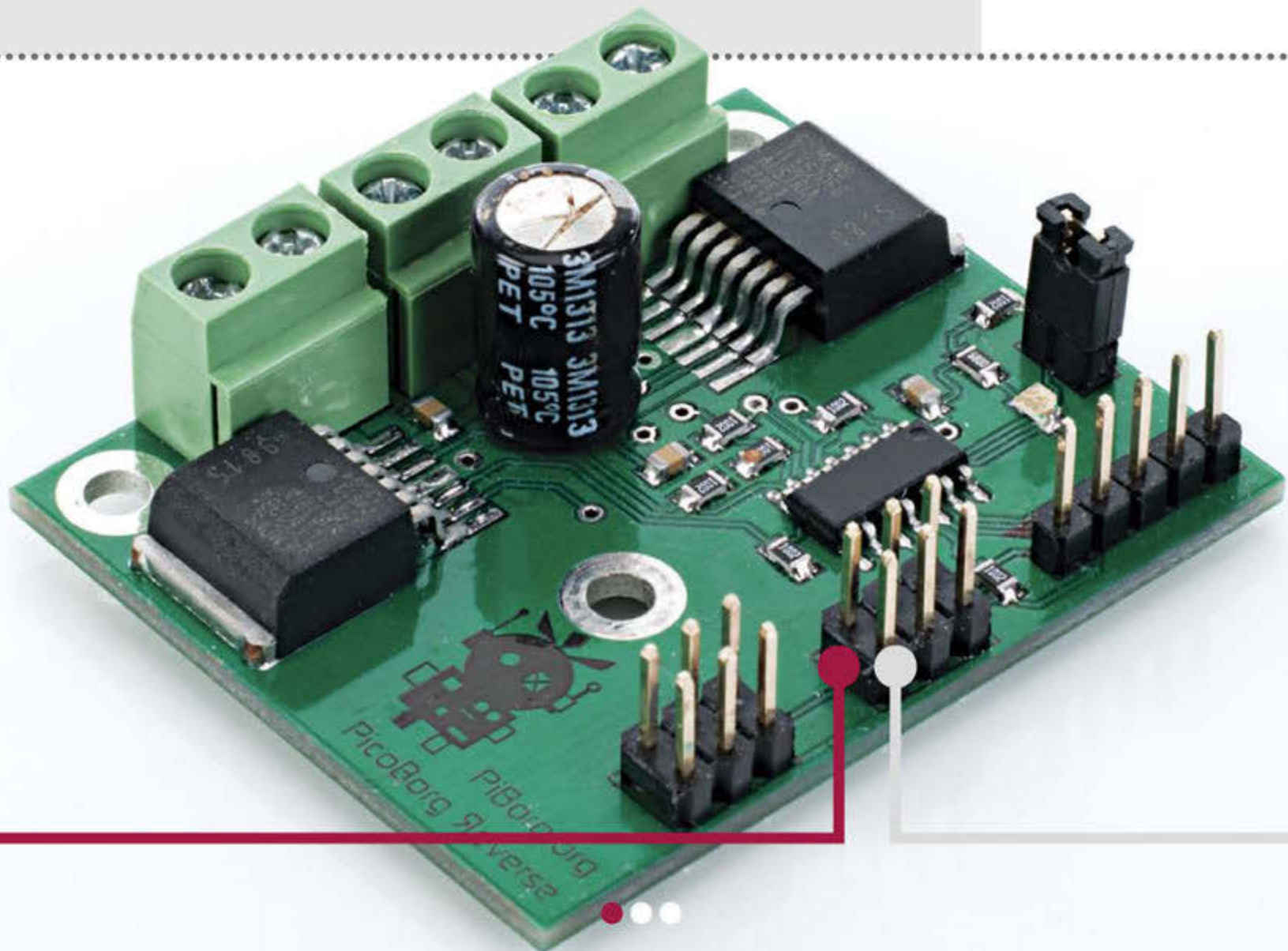


THE PROJECT  
ESSENTIALS

PiBorg PicoBorg  
Reverse

[piborg.org/picoborgrev](http://piborg.org/picoborgrev)

**Below** Remember DoodleBorg from the second issue? There's one of these inside it



easier option, though, is to purchase an add-on board with a suitable motor driver IC already installed. We've chosen to demonstrate the PicoBorg Reverse – a board so versatile they can be daisy-chained together to run a robot powerful enough to tow a caravan...

## Why do it?

Why add motor control to your Raspberry Pi? There's a great range of project ideas, from a motorised dolly to move your camera during time-lapse photography shoots, to autonomous and remote-controlled robots able to scoot around the house. We picked PiBorg's PicoBorg Reverse for a couple of reasons. It caters for the widest range of motor sizes, it doesn't hog the entire GPIO header (a real bug-bear with many other solutions) and it offers its own expansion header so you can daisy-chain boards together and still access the all-important power and ground pins for further use. It's far from being the cheapest motor driver on the market, but it's certainly one of the most project-friendly and user-friendly.

## How to use it

In terms of setting up, the guys at PiBorg have been incredibly thoughtful. The PicoBorg Reverse comes with two colour-coded three-pin cables to connect the board to your Raspberry Pi's GPIO pins. It also comes with a mounting kit so you can easily connect the board to your Raspberry Pi without losing access to any of the connectors. With the board mounted and the two cables connected as per their instructions, the installation process turns to the software side of things. Check out the step-by-step software installation guide on the next page for full details. The Python library supplied with the PicoBorg Reverse is very full-featured and even allows for hardware

**“PicoBorg Reverse caters for the widest range of motor sizes, it doesn't hog the entire GPIO header and it offers its own expansion header”**



immobilisation should there be any difficulties. The library itself also includes a `Help()` text function that breaks down and lists every function and its parameters in plain English.

## PicoBorg Reverse software installation

### 01 Download the examples

Create a folder for the Python module and example scripts to go in. At the terminal type:

```
mkdir ~/picoborgrev
```

Now navigate into the folder (`cd ~/picoborgrev`).

### 02 Run the script

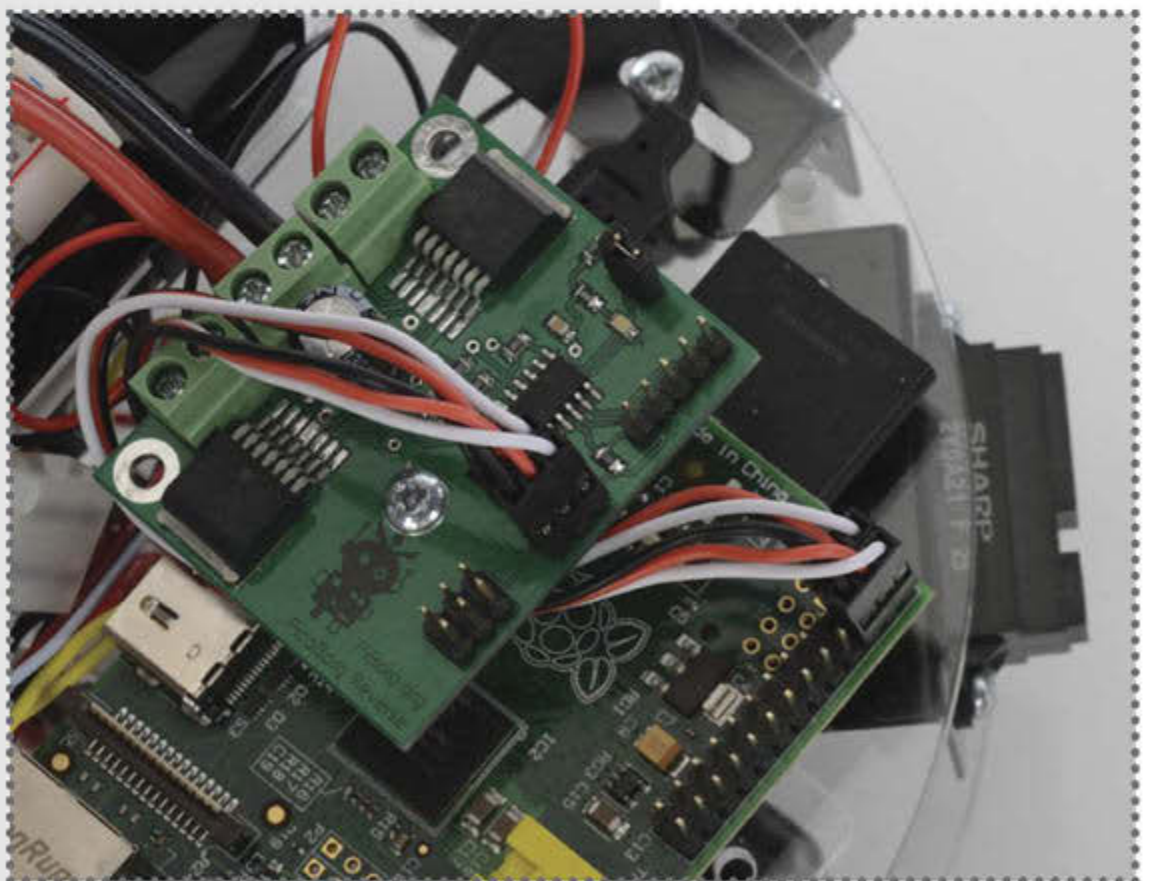
The PicoBorg Reverse uses I2C for communication, but instead of manually setting things up, all you need to do is unzip the download, change the permissions (with `chmod +x install.sh`), then run the script simply by typing `./install.sh` at the command prompt. It really is that straightforward.

### 03 Check out the example programs

Assuming you've got your motors already connected, you're ready to go. You'll even find a new desktop shortcut on your Raspberry Pi with a small GUI program to help you get started. You can find all the example programs and Python library functions at [piborg.org/picoborgrev/examples](http://piborg.org/picoborgrev/examples).

“Instead of manually setting things up, all you need to do is unzip the download, change the permissions, then run the script”

**Below** Just two three-wire connectors are needed between the Pi and the Reverse





# Amplified stereo speakers

Listen to what your Pi has to say without having to plug in earphones



## What is it?

MAX98306 is a small board that connects to a Pi on one end and stereo speakers on the other.

## Why do it?

The Raspberry Pi has no on-board speakers and can only output via HDMI or the analogue audio jack. It's not amplified well, so the output is rather quiet.

## How to use it

Strip the wire on your headphone-style cable. Unravel one side of the cable and attach each part to either the positive (+) and negative (-) of the R or L side. Then, attach the other cable to the remaining side's positive and negative. Wire up the speakers, attach VDD to power and GND to ground, choose a gain on the gain selector pins (start lower at first) and finally plug the jack into the Raspberry Pi.

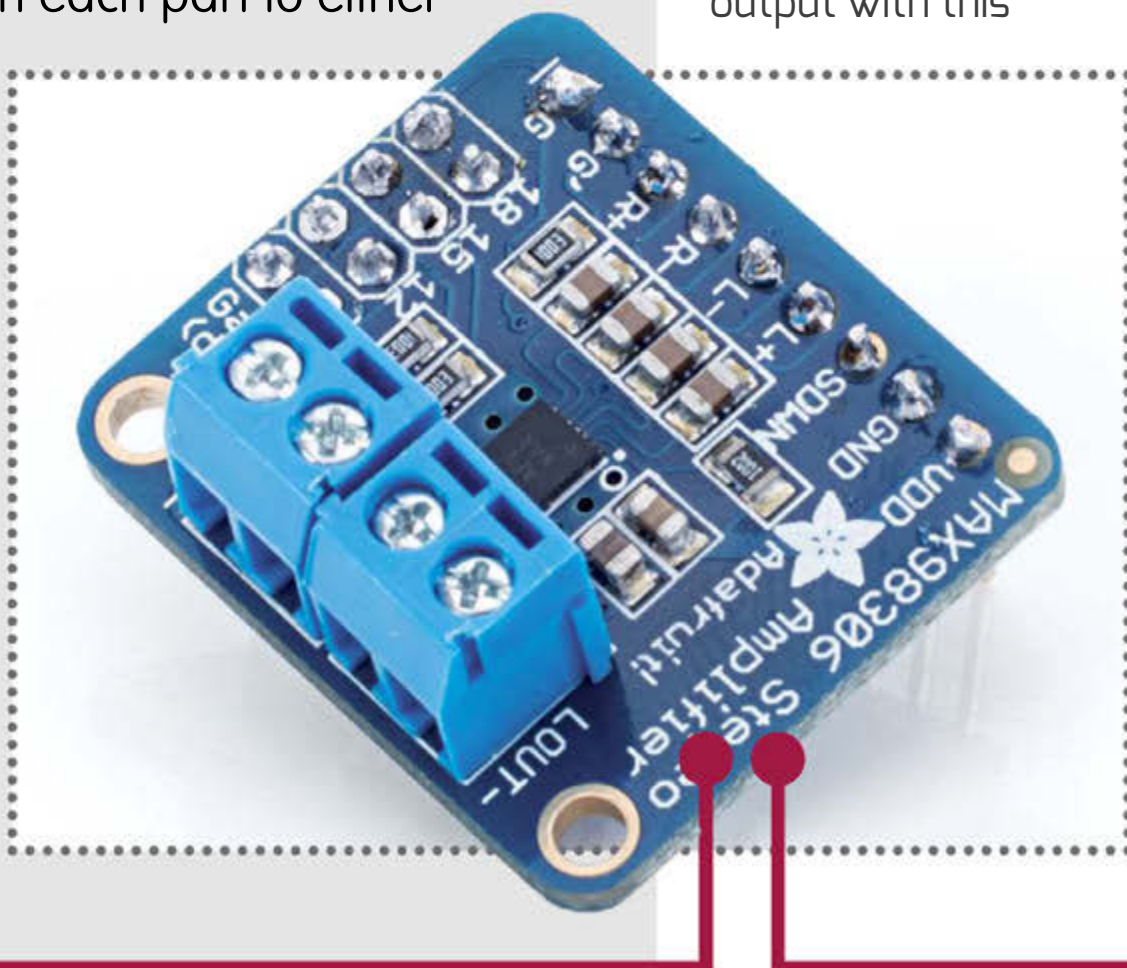


THE PROJECT  
ESSENTIALS

Adafruit Stereo  
Amplifier  
MAX98306

[www.modmypi.com](http://www.modmypi.com)

**Below** Don't let its size fool you – you can get decent audio output with this







# What is the VideoCore graphics core stack?

Broadcom and the Raspberry Pi Foundation opened new code for the Raspberry Pi – but why does this matter?

“There are many types of VideoCoreIV chipsets – the one in the Raspberry Pi is known as BCM2835”



**Q One day I went onto the Raspberry Pi website and I saw that the VideoCoreIV graphics stack is now open to everyone. There are multiple parts of this that I don't quite understand. What's a graphics core – is it like a graphics card or chip?**

**A** In its essence, yes. The Raspberry Pi does not have as many separate processors on the board as a full computer and part of its architecture includes a lot of functions on single chips. Hence the term 'system-on-a-chip', or SoC. The graphics core is part of the larger chip in the centre of the Raspberry Pi and handles the graphical processing for the system.

**Q Okay, so the VideoCoreIV is the brand name or model or something?**

**A** Yes, it's part of the Broadcom chipset that is used in the Raspberry Pi, though they were originally used in mobile phones. There are many types of VideoCoreIV chipsets – the one in the Raspberry Pi is known as BCM2835. The numbers aren't all that important though.

**Q The name Broadcom sounds familiar, where might I have heard of it in relation to the Pi?**

**A** Well as mentioned, Broadcom supplies the SoC used by the Raspberry Pi. You may also have heard or seen that Eben Upton is an employee of Broadcom; he's the main guy behind the Raspberry Pi's creation and is a pretty prominent figure in the The Raspberry Pi Foundation that backs and supports the Pi.

**Q So is the Raspberry Pi Foundation part of Broadcom?**

**A** No. While Eben Upton works for Broadcom and is a founder of the Raspberry Pi Foundation, the Foundation is an independent charity and completely separate to

## Things to expect

These applications can make use of the drivers to increase their performance



### Quake III Arena

This classic PC FPS game used to be a LAN party staple in the time before Steam and Origin. A clone of it already runs on the Raspberry Pi, and a full working port for the open ARM driver was completed earlier this year – install it via [bit.ly/1h4VQlg](http://bit.ly/1h4VQlg).

### Android

Heard of Razdroid? This ambitious port of Android to the Raspberry Pi used CyanogenMod images, however it could only go so far without full access to the VideoCore and the builds are rather unstable. This may not be the project that does it, but someone out there will be able to now create much better Pi ports.





Broadcom. They have a good relationship though; they have an excellent deal on the chips thanks to Eben Upton's involvement.

**Q Is there anything that's particularly special about the VideoCore that they would use it?**

**A** Well it's quite powerful, and has specific in-built code to help decode 1080p content. It's why it can play such high-quality content while still being relatively under-powered compared to similar PCs or systems.

**Q What has actually been made open for the VideoCore, then?**

**A** The actual core itself's documentation and the graphics stack to go along with it. This allows developers full access to the VideoCore, something that is rare for ARM. Traditionally, any communication with an ARM graphics core is done via a blob of code that acts as a driver; this can now be effectively bypassed.

**Q So does this mean the drivers for the VideoCore are open source then?**

**A** Not exactly, because it doesn't use a driver in the traditional sense. The blob of code is a closed-source binary driver that the Linux kernel communicates with to access the VideoCore.

**Q Why has this blob not been made open source?**

**A** The blob is very limited, and the Raspberry Pi would be better served by having proper drivers developed for it.

**Q So why does that matter?**

**A** They can then see how the VideoCore hardware works and how it uses data, allowing the development of an

## Things to expect



### XBMC

The media centre software already runs very well on the Raspberry Pi, however there's always room for improvement and optimisation. For example, slow-down issues with the interface may be reduced as load is spread more evenly throughout the system.

open source driver. This means applications, distros and operating systems can make better use of it as they can be optimised for the VideoCore.

**Q Why wouldn't you do that for all the hardware anyway?**

**A** There's only so much optimisation that you can do when you don't have all of the documentation or access to the hardware, and optimisation is very time-consuming. You couldn't possibly do it for every single hardware type and combination; this is actually one of the benefits of drivers in the first place.

**Q The Raspberry Pi is open source though, right? Why was the graphics stack not?**

**A** The Raspberry Pi hardware is not specifically open source, at least it was never intended to be. Over the past couple of years, more and more of the hardware has been opened up, with this VideoCore documentation being highly requested by the community. It does use open source distros though.

**Q Okay, so the stack is now open source and devs can use them. What will they be using them for?**

**A** To create open source drivers for the Raspberry Pi, among other devices. One of the issues that has stalled any Android porting to the Raspberry Pi has been the closed VideoCore. Now that it's open, it should open up these avenues again. Not only that, but video players, games and other distros can take advantage of it to optimise the software for the Pi. XBMC regularly adds Raspberry Pi-specific features into its codebase and this could mean it works much better on the Raspberry Pi in the future.

“Video players, games and other distros can take advantage of it to optimise software for the Pi”





**Q Well, it all sounds pretty interesting, is it something I should be looking to use?**

**A** Unless you're doing serious distro or porting work, you will likely never need to touch it. That's not really the point of this release though, as it will positively affect your Raspberry Pi experience in the long term.

**Q Just how far along are they with the open source drivers then?**

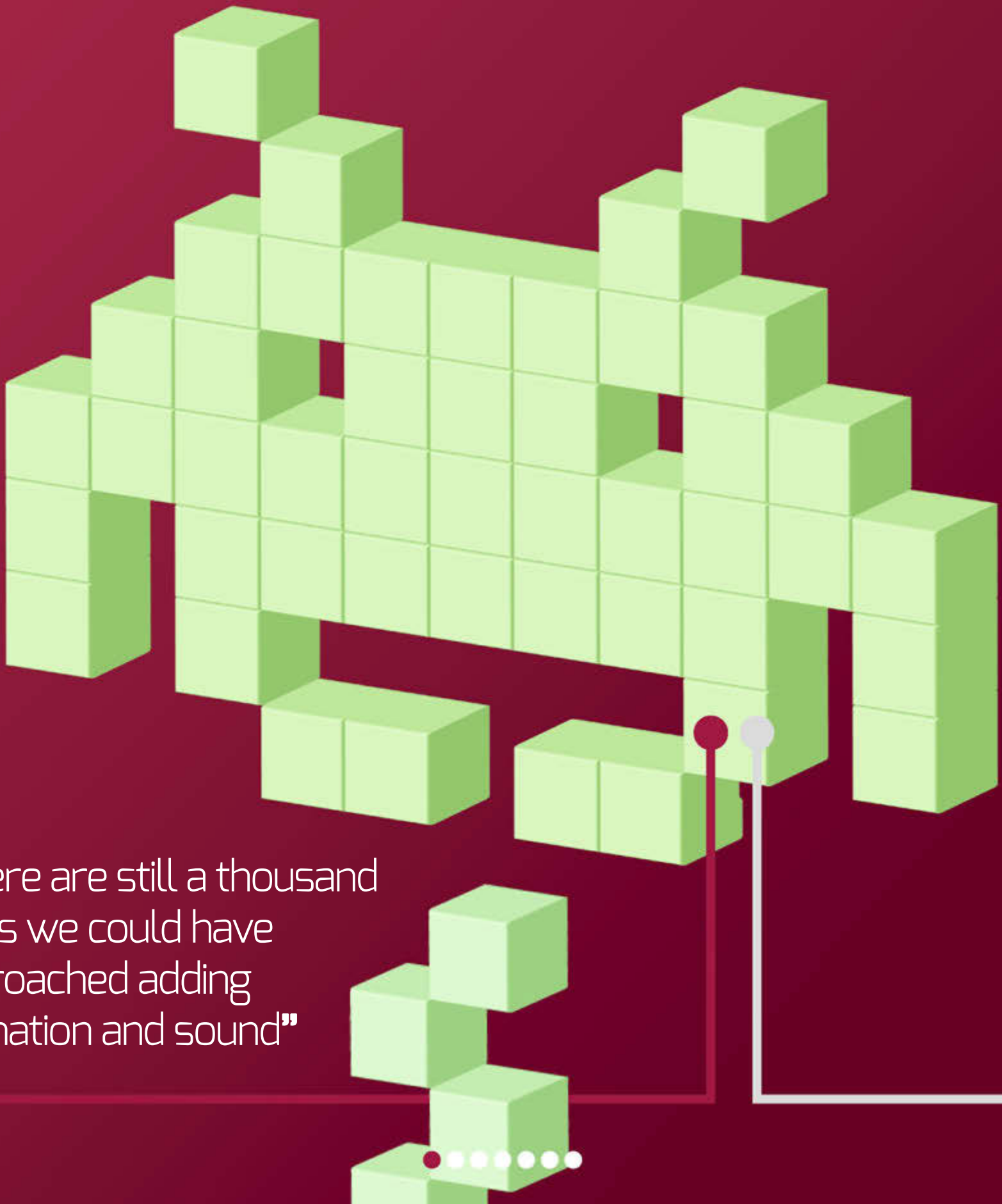
**A** Good news, the drivers have basically been created and a custom kernel that includes them has been released. This was spurred on by the Raspberry Pi Foundation, who created a \$10,000 bounty for the first person to show Quake III running natively on the Raspberry Pi. To do this, you would need to port the released code to the Raspberry Pi and create a rudimentary driver, which will now be used in a full open source drive project. The winner, Simon Hall, managed to do it in just over a month.

“The drivers have basically been created and a custom kernel that includes them has been released”



# Add animation and sound to Pivaders

Last issue we wrote a Space Invaders clone in just 300 lines of Python. Now we'll add SFX and improve graphics



“There are still a thousand ways we could have approached adding animation and sound”





We had great fun creating our basic Space Invaders clone for issue 4. Pygame's ability to group, manage and detect collisions thanks to the Sprite class really made a great difference to our project, not just in terms of length but in simplicity. Even working within the clearly defined framework Pygame offers, there are still a thousand ways we could have approached adding animation and sound. We could have created any one of a dozen classes to create and manage containers of individual images, or read in a sprite sheet (a single image full of smaller, separate images) which we could then draw (or blit) to the screen. For the sake of simplicity and performance, we integrated a few animation methods into our Game class and opted to use a sprite sheet. Not only does it make it very easy to draw to the screen, but it also keeps the asset count under control and keeps performance levels up, which is especially important for the Pi.



## THE PROJECT ESSENTIALS

### Latest Raspbian Image

[raspberrypi.org/  
downloads](https://raspberrypi.org/downloads)

### Python

[python.org/doc](https://python.org/doc)

### Pygame

[pygame.org/docs](https://pygame.org/docs)

### Art assets

[opengameart.org](https://opengameart.org)

## 01 Setting up dependencies

As we recommended with last issue's tutorial, you'll get much more from the exercise if you download the code ([git.io/8QsK-w](https://git.io/8QsK-w)) and use it for reference as you create your own animations and sound for your Pygame projects. Regardless of whether you just want to simply preview and play or walk-through the code to get a better understanding of basic game creation, you're still going to need to satisfy some basic dependencies. The two key requirements here are Pygame and Git, both of which are installed by default on up-to-date Raspbian installations. If you're unsure if you have them, though, type the following at the command line:

```
sudo apt-get install python-pygame git
```

## Skills to learn

### Python

You'll need a basic grounding in Python to follow along with the code. Try [codecademy.com](https://www.codecademy.com)

### Classes

Hopefully the article shows how useful classes can be to create and manipulate multiple objects with ease.



## 02 Downloading pivaders

Git is a superb version control solution that helps programmers safely store their code and associated files. Not only does it help you retain a full history of changes, it means you can 'clone' entire projects to use and work on from places like **github.com**. To clone the version of the project we created for this tutorial, go to your home folder from the command line (`cd ~`) and type:

```
git pull https://github.com/LinuxUserMag/  
pivaders.git
```

This will create a folder called **pivaders** – go inside (`cd pivaders`) and take a look around.

## 03 Navigating the project

The project is laid out quite simply across a few subfolders. Within **pivaders** sits a licence, readme and a second **pivaders** folder. This contains the main game file, **pivaders.py**, which launches the application. Within the **data** folder you'll find subfolders for both graphics and sound assets, as well as the font we've used for the title screen and scores. To take pivaders for a test-drive, simply enter the pivaders subdirectory (`cd pivaders/  
pivaders`) and type:

```
python pivaders.py
```

Use the arrow keys to steer left and right and the space bar to shoot. You can quit to the main screen with the Esc key and press it again to exit the game completely.

## 04 Animation & sound

Compared with the game from last month's tutorial, you'll see it's now a much more dynamic project. The protagonist ship now leans into the turns as you change direction and corrects itself when you either press the opposite direction or lift your finger off the button. When you shoot an alien ship, it explodes with

“Within the data folder you'll find subfolders for both graphics and sound assets, as well as the font we've used for the title screen and scores”



several frames of animation and should you take fire, a smaller explosion occurs on your ship. Music, lasers and explosion sound effects also accompany the animations as they happen.

## 05 Finding images to animate

Before we can program anything, it's wise to have assets set up in a way we can use them. As mentioned, we've opted to use sprite sheets; these can be found online or created with GIMP with a little practice. Essentially they're a mosaic made up of individual 'frames' of equally sized and spaced images representing each frame. Find ready-made examples at **opengameart.org**, as used here.

“Sprite sheets are essentially a mosaic made up of individual ‘frames’ of equally sized and spaced images representing each frame”

## 06 Tweaking assets

While many of the assets on sites like **opengameart.org** can be used as is, you may want to import them into an image-editing application like GIMP to configure them to suit your needs – as we did with our ship sheet asset to help us keep the code simple. We started with the central ship sprite and centred it into a new window. We set the size and width of the frame and then copy-pasted the other frames either side of it. We ended up with 11 frames of exactly the same size and width in a single document. Pixel-perfect precision on size and width is key, so we can just multiply it to find the next frame.

## 07 Loading the sprite sheet

Since we're inheriting from the `Sprite` class to create our **Player** class, we can easily alter how the player looks on screen by changing **Player.image**. First, we need to load our ship sprite sheet with **pygame.image.load()**. Since we made our sheet with a transparent

background, we can append **.convert\_alpha()** to the end of the line so the ship frames render correctly (without any background). We then use **subsurface** to set the initial **Player.image** to the middle ship sprite on the sheet. This is set by **self.ani\_pos**, which has an initial value of 5. Changing this value will alter the ship image drawn to the screen: '0' would draw it leaning fully left, '11' fully to the right.

## 08 Animation flags

Slightly further down the list in the initialising code for the **Game** class, we also set two flags for our player animation: **self.animate\_left** and **self.animate\_right**. As you'll see in the **Control** method of our **Game** class, we use these to 'flag' when we want animations to happen using Boolean (**True** or **False**) values. It also allows us to 'automatically' animate the player sprite back to its natural resting state (otherwise the ship will continue to look as if it's flying left when it has stopped).

## 09 The animation method

These flags pop up again in the core animation code for the player: **animate\_player()** within the **Game** class. Here we use nested if statements to control the animation and set the player image accordingly. Essentially it states that if the **animate\_right** flag is **True** and if the current animation position is different to what we want, we incrementally increase the **ani\_pos** variable and set the player's image accordingly. The **Else** statement then animates the ship sprite back to its resting state and the same logic is then applied in the opposite direction.

“If the current animation position is different to what we want, we incrementally increase the **ani\_pos** variable and set the player's image accordingly”



## 10 Animating explosions

The **player\_explosion()** and **alien\_explosion()** methods that come after the player animation block in the **Game** class are similar but simpler executions of essentially the same thing. As we only need to run through the same predefined set of frames (this time vertically), we only need to see if the **self.explode** and **self.alien\_explode** flags are **True** before we increment the variables that change the image displayed. As the sprite sheet is vertical, the variables **alien\_explode\_pos** and **explosion\_image** are set to a different part of **subsurface** than before.

“Just obtain a suitable piece of music in your preferred format and load it using the Mixer Pygame class”

## 11 Adding music to your project

Pygame makes it easy to add a musical score to a project. Just obtain a suitable piece of music in your preferred format (we found ours via **freemusicarchive.org**) and load it using the **Mixer** Pygame class. As it's already been initialised via **pygame.init()**, we can go ahead and load the music with this code:

```
pygame.mixer.music.load('data/sound/10_
    Arpanauts.ogg')
pygame.mixer.music.play(-1)
pygame.mixer.music.set_volume(0.7)
```

The **music.play(-1)** requests that the music should start with the app and continue to loop until it quits. If we replaced -1 with 5, the music would loop five times before ending. Learn more about the Mixer class via [www.pygame.org/docs/ref/mixer.html](http://www.pygame.org/docs/ref/mixer.html).

## 12 Using sound effects

Loading and using sounds is similar to how we do so for images in Pygame. First we load the sound effect

using a simple assignment. For the laser beam, the initialisation looks like this:

```
self.bullet_fx = pygame.mixer.Sound('data/sound/medetix__pc-bitcrushed-lazer-beam.ogg')
```

Then we simply trigger the sound effect at the appropriate time. With the laser, we want it to play whenever we press the space bar to shoot, so we place it in the **Game** class's Control method, straight after we raise the **shoot\_bullet** flag.

If you're struggling to find free and open sound effects, we recommend **www.freesound.org**.

“With the laser, we place it in the Game class's Control method after we raise shoot\_bullet”

**Below** The Freesound site is a good place to find free and open sound effects for your projects

The screenshot shows the Freesound website interface. At the top, there's a navigation bar with links for Home, Sounds, Forums, People, and Help. A search bar is also present. The main content area features a user profile for 'medetix', including a placeholder for a profile picture and a list of statistics: 'Send a private message to this user', 'Has been a user for 1 year, 1 month', 'Number of sounds: 9', 'Sounds downloaded by medetix', 'Packs downloaded by medetix', and 'All comments on medetix's sounds'. Below the profile, there's a section titled 'medetix's latest sounds' which lists three sound files: 'SD White Noise Snare.wav', 'PC Coin.wav', and 'PC Quick Lazer.wav'. Each entry includes a play button, a waveform visualization, a description, and download statistics. To the right of this section, there's a 'medetix's most used tags' section listing tags like 'Beam Bit', 'Crusher', 'Daft Derezzed', 'EDM', 'Lazer Percussion', and 'Punk'.

Register Log In Upload Sounds

search sounds

Home Sounds Forums People Help

### medetix

- Send a private message to this user
- Has been a user for 1 year, 1 month
- Number of sounds: 9
- Sounds downloaded by medetix
- Packs downloaded by medetix
- All comments on medetix's sounds

#### medetix's latest sounds

Sound File	Description	Downloads	Comments
SD White Noise Snare.wav	That snare drum that was in that Tron movie (Daft Punk - Derezzed)	66 downloads	0 comments
PC Coin.wav	A pixel coin tossed sound sorta. That would fit nicely with some EDM production	67 downloads	0 comments
PC Quick Lazer.wav	A quick lazer beam that would fit nicely with some EDM production	216 downloads	1 comment

#### medetix's most used tags

Beam Bit Crusher Daft Derezzed EDM Lazer Percussion Punk



# The Code

## ADD ANIMATION AND SOUND TO PIVADERS

```
class Game(object):
    def __init__(self):
        pygame.init()
        pygame.font.init()
        self.clock = pygame.time.Clock()
        self.game_font = pygame.font.Font(
            'data/Orbitracer.ttf', 28)
        self.intro_font = pygame.font.Font(
            'data/Orbitracer.ttf', 72)
        self.screen = pygame.display.set_mode([RES[0], RES[1]])
        self.time = pygame.time.get_ticks()
        self.refresh_rate = 20; self.rounds_won = 0
        self.level_up = 50; self.score = 0
        self.lives = 2
        self.player_group = pygame.sprite.Group()
        self.alien_group = pygame.sprite.Group()
        self.bullet_group = pygame.sprite.Group()
        self.missile_group = pygame.sprite.Group()
        self.barrier_group = pygame.sprite.Group()
        self.all_sprite_list = pygame.sprite.Group()
        self.intro_screen = pygame.image.load(
            'data/graphics/start_screen.jpg').convert()
        self.background = pygame.image.load(
            'data/graphics/Space-Background.jpg').convert()
        pygame.display.set_caption('Pivaders - ESC to exit')
        pygame.mouse.set_visible(False)
        Alien.image = pygame.image.load(
            'data/graphics/Spaceship16.png').convert()
        Alien.image.set_colorkey(WHITE)
        self.ani_pos = 5 # 11 images of ship
```

“First, we need to load our ship sprite sheet with `pygame.image.load()`. We can append `.convert_alpha()` to the end of the line so the ship frames render correctly”

# The Code

## ADD ANIMATION AND SOUND TO PIVADERS

```
self.ship_sheet = pygame.image.load(
    'data/graphics/ship_sheet_final.png').convert_alpha()
Player.image = self.ship_sheet.subsurface(
    self.ani_pos*64, 0, 64, 61)
self.animate_right = False
self.animate_left = False
self.explosion_sheet = pygame.image.load(
    'data/graphics/explosion_new1.png').convert_alpha()
self.explosion_image = self.explosion_sheet.subsurface(0, 0, 79, 96)
self.alien_explosion_sheet = pygame.image.load(
    'data/graphics/alien_explosion.png')
self.alien_explode_graphics = self.alien_explosion_sheet. subsurface(0, 0, 94, 96)
self.explode = False
self.explode_pos = 0; self.alien_explode = False
self.alien_explode_pos = 0
pygame.mixer.music.load('data/sound/10_Arpanauts.ogg')
pygame.mixer.music.play(-1)
pygame.mixer.music.set_volume(0.7)
self.bullet_fx = pygame.mixer.Sound(
    'data/sound/medetix__pc-bitcrushed-lazer-beam.ogg')
self.explosion_fx = pygame.mixer.Sound(
    'data/sound/timgormly__8-bit-explosion.ogg')
self.explosion_fx.set_volume(0.5)
self.explodey_alien = []
GameState.end_game = False
GameState.start_screen = True
GameState.vector = 0
GameState.shoot_bullet = False
```

```
def control(self):
    for event in pygame.event.get():
```

### ship\_sheet

We set the player image to equal one small segment of the sprite sheet using 'ani\_pos'. Change it to change the picture



# The Code

## ADD ANIMATION AND SOUND TO PIVADERS

```
if event.type == pygame.QUIT:
    GameState.start_screen = False
    GameState.end_game = True
if event.type == pygame.KEYDOWN \
and event.key == pygame.K_ESCAPE:
    if GameState.start_screen:
        GameState.start_screen = False
        GameState.end_game = True
        self.kill_all()
    else:
        GameState.start_screen = True
self.keys = pygame.key.get_pressed()
if self.keys[pygame.K_LEFT]:
    GameState.vector = -1
    self.animate_left = True
    self.animate_right = False
elif self.keys[pygame.K_RIGHT]:
    GameState.vector = 1
    self.animate_right = True
    self.animate_left = False
else:
    GameState.vector = 0
    self.animate_right = False
    self.animate_left = False

if self.keys[pygame.K_SPACE]:
    if GameState.start_screen:
        GameState.start_screen = False
        self.lives = 2
        self.score = 0
        self.make_player()
```

### Set flags

We've added 'animate\_left' and 'animate\_right' Boolean flags to the control method. When they're true, the actual animation code is called via a separate method.



# The Code

## ADD ANIMATION AND SOUND TO PIVADERS

```
self.make_defenses()
self.alien_wave(0)
else:
    GameState.shoot_bullet = True
    self.bullet_fx.play()
```

```
def animate_player(self):
    if self.animate_right:
        if self.ani_pos < 10:
            Player.image = self.ship_sheet.subsurface(
                self.ani_pos*64, 0, 64, 61)
            self.ani_pos += 1
        else:
            if self.ani_pos > 5:
                self.ani_pos -= 1
            Player.image = self.ship_sheet.subsurface(
                self.ani_pos*64, 0, 64, 61)
```

```
if self.animate_left:
    if self.ani_pos > 0:
        self.ani_pos -= 1
        Player.image = self.ship_sheet.subsurface(
            self.ani_pos*64, 0, 64, 61)
    else:
        if self.ani_pos < 5:
            Player.image = self.ship_sheet.subsurface(
                self.ani_pos*64, 0, 64, 61)
            self.ani_pos += 1
```

```
def player_explosion(self):
    if self.explode:
```

### fx.play()

Having already loaded the sound effect we want when we shoot, we now just need to call it when we press the space bar.





# The Code

## ADD ANIMATION AND SOUND TO PIVADERS

```
if self.explode_pos < 8:
    self.explosion_image = self.explosion_sheet.subsurface(0, self.explode_pos*96, 79, 96)
    self.explode_pos += 1
    self.screen.blit(self.explosion_image, [self.player.rect.x - 10, self.player.rect.y - 30])
else:
    self.explode = False
    self.explode_pos = 0

def alien_explosion(self):
    if self.alien_explode:
        if self.alien_explode_pos < 9:
            self.alien_explode_graphics = self.alien_explosion_sheet.subsurface(0, self.
alien_explode_pos*96, 94, 96)
            self.alien_explode_pos += 1
            self.screen.blit(self.alien_explode_graphics, [int(self.explodey_alien[0]) - 50 ,
int(self.explodey_alien[1]) - 60])
        else:
            self.alien_explode = False
            self.alien_explode_pos = 0
            self.explodey_alien = []

def splash_screen(self):
    while GameState.start_screen:
        self.kill_all()
        self.screen.blit(self.intro_screen, [0, 0])
        self.screen.blit(self.intro_font.render(
"PIVADERS", 1, WHITE), (265, 120))
        self.screen.blit(self.game_font.render(
"PRESS SPACE TO PLAY", 1, WHITE), (274, 191))
        pygame.display.flip()
        self.control()
```

# The Code

## ADD ANIMATION AND SOUND TO PIVADERS

```
self.clock.tick(self.refresh_rate / 2)
```

```
def make_player(self):
```

```
    self.player = Player()
```

```
    self.player_group.add(self.player)
```

```
    self.all_sprite_list.add(self.player)
```

```
def refresh_screen(self):
```

```
    self.all_sprite_list.draw(self.screen)
```

```
    self.animate_player()
```

```
    self.player_explosion()
```

```
    self.alien_explosion()
```

```
    self.refresh_scores()
```

```
    pygame.display.flip()
```

```
    self.screen.blit(self.background, [0, 0])
```

```
    self.clock.tick(self.refresh_rate)
```

```
def refresh_scores(self):
```

```
    self.screen.blit(self.game_font.render(
```

```
        "SCORE " + str(self.score), 1, WHITE), (10, 8))
```

```
    self.screen.blit(self.game_font.render(
```

```
        "LIVES " + str(self.lives + 1), 1, RED), (355, 575))
```

```
def alien_wave(self, speed):
```

```
    for column in range(BARRIER_COLUMN):
```

```
        for row in range(BARRIER_ROW):
```

```
            alien = Alien()
```

```
            alien.rect.y = 65 + (column * (
```

```
                ALIEN_SIZE[1] + ALIEN_SPACER))
```

```
            alien.rect.x = ALIEN_SPACER + (
```

```
                row * (ALIEN_SIZE[0] + ALIEN_SPACER))
```

“The player\_explosion() and alien\_explosion() methods are similar but simpler executions of essentially the same thing”



# The Code

## ADD ANIMATION AND SOUND TO PIVADERS

```
self.alien_group.add(alien)
self.all_sprite_list.add(alien)
alien.speed -= speed
```

```
def make_bullet(self):
    if GameState.game_time - self.player.time > self.player.speed:
        bullet = Ammo(BLUE, BULLET_SIZE)
        bullet.vector = -1
        bullet.speed = 26
        bullet.rect.x = self.player.rect.x + 28
        bullet.rect.y = self.player.rect.y
        self.bullet_group.add(bullet)
        self.all_sprite_list.add(bullet)
        self.player.time = GameState.game_time
        GameState.shoot_bullet = False
```

```
def make_missile(self):
    if len(self.alien_group):
        shoot = random.random()
        if shoot <= 0.05:
            shooter = random.choice([
                alien for alien in self.alien_group])
            missile = Ammo(RED, MISSILE_SIZE)
            missile.vector = 1
            missile.rect.x = shooter.rect.x + 15
            missile.rect.y = shooter.rect.y + 40
            missile.speed = 10
            self.missile_group.add(missile)
            self.all_sprite_list.add(missile)
```

```
def make_barrier(self, columns, rows, spacer):
```



# The Code

## ADD ANIMATION AND SOUND TO PIVADERS

```
for column in range(columns):
    for row in range(rows):
        barrier = Block(WHITE, (BLOCK_SIZE))
        barrier.rect.x = 55 + (200 * spacer) + (row * 10)
        barrier.rect.y = 450 + (column * 10)
        self.barrier_group.add(barrier)
        self.all_sprite_list.add(barrier)

def make_defenses(self):
    for spacing, spacing in enumerate(xrange(4)):
        self.make_barrier(3, 9, spacing)

def kill_all(self):
    for items in [self.bullet_group, self.player_group,
self.alien_group, self.missile_group, self.barrier_group]:
        for i in items:
            i.kill()

def is_dead(self):
    if self.lives < 0:
        self.screen.blit(self.game_font.render(
            "The war is lost! You scored: " + str(
self.score), 1, RED), (250, 15))
        self.rounds_won = 0
        self.refresh_screen()
        self.level_up = 50
        self.explode = False
        self.alien_explode = False
        pygame.time.delay(3000)
        return True
```

“As we only need to run through the same predefined set of frames, we only need to see if the self.explode and self.alien\_explode flags are True before we increment the variables”





# The Code

## ADD ANIMATION AND SOUND TO PIVADERS

```
def defenses_breached(self):
    for alien in self.alien_group:
        if alien.rect.y > 410:
            self.screen.blit(self.game_font.render(
                "The aliens have breached Earth defenses!",
                1, RED), (180, 15))
            self.refresh_screen()
            self.level_up = 50
            self.explode = False
            self.alien_explode = False
            pygame.time.delay(3000)
            return True
```

```
def win_round(self):
    if len(self.alien_group) < 1:
        self.rounds_won += 1
        self.screen.blit(self.game_font.render(
            "You won round " + str(self.rounds_won) +
            " but the battle rages on", 1, RED), (200, 15))
        self.refresh_screen()
        pygame.time.delay(3000)
        return True
```

```
def next_round(self):
    self.explode = False
    self.alien_explode = False
    for actor in [self.missile_group,
                  self.barrier_group, self.bullet_group]:
        for i in actor:
            i.kill()
    self.alien_wave(self.level_up)
```



# The Code

## ADD ANIMATION AND SOUND TO PIVADERS

```
self.make_defenses()
```

```
self.level_up += 50
```

```
def calc_collisions(self):
```

```
    pygame.sprite.groupcollide(
```

```
        self.missile_group, self.barrier_group, True, True)
```

```
    pygame.sprite.groupcollide(
```

```
        self.bullet_group, self.barrier_group, True, True)
```

```
    for z in pygame.sprite.groupcollide(
```

```
        self.bullet_group, self.alien_group, True, True):
```

```
        self.alien_explode = True
```

```
        self.explodey_alien.append(z.rect.x)
```

```
        self.explodey_alien.append(z.rect.y)
```

```
        self.score += 10
```

```
        self.explosion_fx.play()
```

```
    if pygame.sprite.groupcollide(
```

```
        self.player_group, self.missile_group, False, True):
```

```
        self.lives -= 1
```

```
        self.explode = True
```

```
        self.explosion_fx.play()
```

```
def main_loop(self):
```

```
    while not GameState.end_game:
```

```
        while not GameState.start_screen:
```

```
            GameState.game_time = pygame.time.get_ticks()
```

```
            GameState.alien_time = pygame.time.get_ticks()
```

```
            self.control()
```

```
            self.make_missile()
```

```
            self.calc_collisions()
```





# The Code

## ADD ANIMATION AND SOUND TO PIVADERS

```
self.refresh_screen()
if self.is_dead() or self.defenses_breached():
    GameState.start_screen = True
for actor in [self.player_group, self.bullet_group,
self.alien_group, self.missile_group]:
    for i in actor:
        i.update()
if GameState.shoot_bullet:
    self.make_bullet()
if self.win_round():
    self.next_round()
self.splash_screen()
pygame.quit()

if __name__ == '__main__':
    pv = Game()
    pv.main_loop()
```

“If you’re struggling to find free and open sound effects, we recommend <http://www.freesound.org>”



# Build an XLoBorg tilt controller

Use the accelerometer in PiBorg's small and affordable XLoBorg device to turn your Pi into a game controller...



"XLoBorg is a board designed to help you measure movement and determine direction, among other things"



If you saw the second issue of **RasPi** then you might remember the DoodleBorg, a massive remote-controlled tank of a vehicle designed and built by PiBorg, makers of add-ons for the Raspberry Pi. This month we're looking at their XLoBorg, a board designed to help you measure movement and determine direction, among other things.

At under £10/\$16, it's a bargain too, because as well as featuring a three-axis accelerometer, it's also kitted out with a three-axis magnetometer (digital compass). In this tutorial we'll be using the accelerometer to turn our Pi into a tilt controller and mock up a simple demo to show how you could integrate it into your Pygame-powered games...



**THE PROJECT  
ESSENTIALS**

**Python 2.7**

[python.org/doc](http://python.org/doc)

**PiBorg XLoBorg**

[piborg.org/xloborg](http://piborg.org/xloborg)

**Pygame**

[pygame.org/docs](http://pygame.org/docs)

## 01 Install the XLoBorg

Installing the XLoBorg onto your Raspberry Pi really couldn't be easier. It simply slots directly over your GPIO pins with the main body of the board facing over your Raspberry Pi. PiBorg has made the software side of the installation super easy too, by creating an installation script that automates the process of installing the required I2C drivers and library files on your Pi.

## 02 Download the software

Make a new directory for the project files in your home folder by typing: `mkdir ~/xloborg`, enter the directory (with `cd ~/xloborg`) and download the package with: `wget www.piborg.org/downloads/xloborg/examples.zip`. Now unzip the package with `unzip examples.zip` and make the `install.sh` script executable with `chmod +x install.sh`. Finally, run the script with `./install.sh` and reboot your Pi once the installation has finished.

“As well as featuring a three-axis accelerometer, it's also kitted out with a three-axis magnetometer”

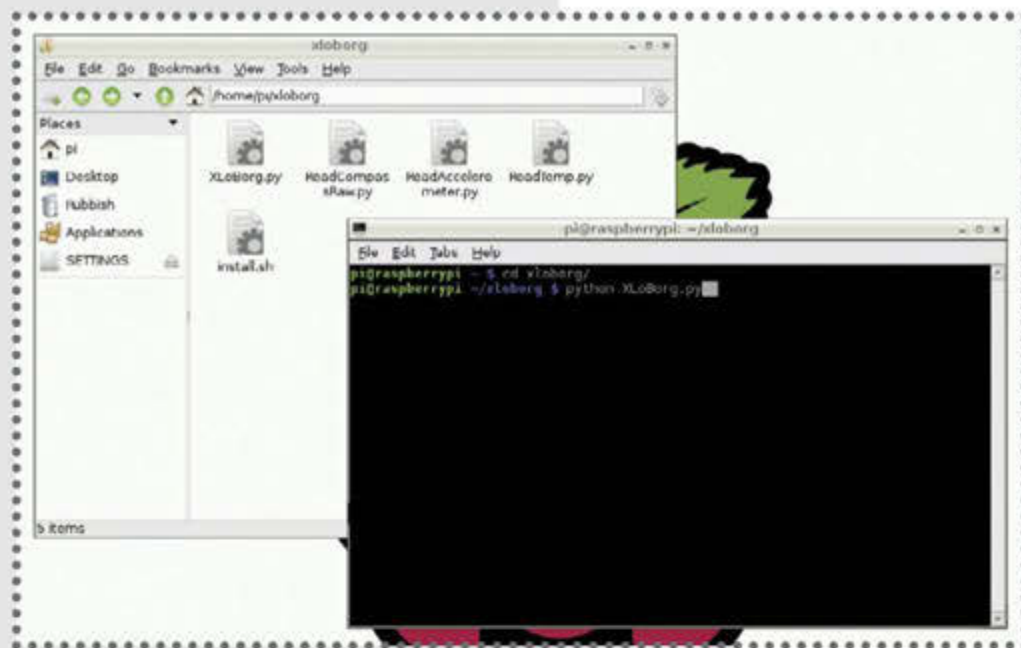




## 03 Inspect and test the library

Whenever you're dealing with a new piece of hardware or a software library designed to abstract a particular process, it's worth looking at the library directly to see how it works. The XLoBorg's library is XLoBorg.py and it contains well-commented code that demonstrates

the main functionality on the board well. The library is also executable with a simple routine to print the sensor results to the terminal. Run it with `python XLoBorg.py` from the terminal to ensure everything's working properly. Move your Pi around to ensure the accelerometer readings change. If not, see the troubleshooting page at [www.piborg.com/xloborg/troubleshoot](http://www.piborg.com/xloborg/troubleshoot).



**Above** Read through the example scripts – they're easy to follow

## 04 Start the script

All being well, you're seeing read-outs from running XLoBorg.py that change when you move the Raspberry Pi around. Now we know everything works, we can start our script with the core XLoBorg functionality. Open a new text editor file and save it as xloborg\_app.py. First we need to initialise the XLoBorg in our script with `XLoBorg.Init()`, then we're going to suppress the print function to save us wading through terminal output with a new line:

```
XLo.Borg.printprintFunction = XLoBorg.NoPrint
```

All we need to do now is create a variable that will carry the accelerometer readings. We've done this with the following line:

“All being well, you're seeing read-outs from running XLoBorg.py that change when you move the Pi around”



```
tilt = XLoBorg.ReadAccelerometer()
```

This will create a Tuple called tilt that contains x, y, z forces (in that order). We don't need the z forces for this example, so we'll inject the readings into a handy little function for our demo app by calling tilt[0] and tilt[1] for the x and y readings respectively.



## 05 Pygame example app

Believe it or not, that's all the XLoBorg code we need to turn our Raspberry Pi into a tilt controller. Next we'll create a bare-bones graphical demo that could easily be turned into a marble tilt game or something similar. At the top of the script we've initiated some constants to cater for the screen size, ball size and colours. Then we initialise Pygame, creating a screen and setting a window caption. The main bulk of the program is a simple Ball class. The two methods update(x, y) and collide() ensure the ball position can be updated and the latter dictates what happens

**Above** Your XLoBorg output should be like this – x and y values

when the ball collides with the edge of the screen. We've constructed the `update()` method to take two variables – this is where we'll inject our accelerometer values during the main loop of the script.

## 06 The main loop

The main loop is designed to enter an infinite loop using the variable `game_over`. While `game_over` is False everything after the `while not game_over:` line will be repeated 30 times every second until we quit it with the Escape key or Ctrl+C. Before we start this loop, though, we ensure `game_over` is indeed False and construct a ball. Once we enter the infinite loop we read the XLoBorg's accelerometer, paint the screen black and call the `ball.update(x, y)` method using the tilt variable to pass the new readings to our ball. We then check the ball's location to ensure it's not trying to leave the screen, then we blit (or draw) the ball on the screen and redraw it using Pygame's `flip()` function.

Once it's complete, save the script and run it with `python xloborg_app.py`. Tilting your Raspberry Pi will move the ball on the screen – if the ball is moving in the wrong directions, simply 'minus' the respective reading as we've done in the example code on the right to flip them.

“We would recommend using a random address range in case you ever want to route to other networks”



# The Code

ADD TILT CONTROL

```
#!/bin/bash/env python
```

```
import pygame
import XLoBorg
```

```
WIDTH = 800
HEIGHT = 600
REF_RATE = 30
BALL_SIZE = (24, 24)
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
```

```
pygame.init()
XLoBorg.Init()
XLoBorg.printprintFunction = XLoBorg.NoPrint
```

```
screen = pygame.display.set_mode([WIDTH, HEIGHT])
clock = pygame.time.Clock()
pygame.display.set_caption('XLoBorg test - Press ESC to quit')
```

```
class Ball(pygame.sprite.Sprite):
    def __init__(self, width, height):
        pygame.sprite.Sprite.__init__(self)
        self.size = (width, height)
        self.image = pygame.Surface([width, height])
        self.image.fill(WHITE)
        self.rect = self.image.get_rect()
        self.rect.x, self.rect.y = WIDTH / 2, HEIGHT / 2
        self.speed = (50, 50)
        self.tilt = [0, 0]
```

## Import and initialise

To start using the XLoBorg in your Python scripts, import the library and call `init()`. We've also suppressed printing to streamline our app



# The Code

## ADD TILT CONTROL

```
def update(self, x, y):
    self.rect.x += x * self.speed[0]
    self.rect.y += y * self.speed[1]
    print 'X =', x, 'Y =', y

def collide(self):
    if self.rect.x > WIDTH - self.size[0]:
        self.rect.x = WIDTH - self.size[0]
    elif self.rect.x < 0:
        self.rect.x = 0
    if self.rect.y > HEIGHT - self.size[1]:
        self.rect.y = HEIGHT - self.size[1]
    elif self.rect.y < 0:
        self.rect.y = 0
```

```
def main_loop():
    game_over = False
    ball = Ball(50, 50)
```

```
while not game_over:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            game_over = True
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                game_over = True
```

```
tilt = XLoBorg.ReadAccelerometer()
screen.fill(BLACK)
ball.update(-tilt[0], tilt[1])
ball.collide()
```

## Update

The update method in the ball class makes the ball move. The method requests an x and y value – we'll be serving it our XLoBorg readings in the main loop later



# The Code

ADD TILT CONTROL

```
screen.blit(ball.image, ball.rect)
pygame.display.flip()
clock.tick(REF_RATE)
pygame.quit()

if __name__ == '__main__':
    try:
        main_loop()

    except KeyboardInterrupt:
        pygame.quit()
```

## Feedback loop

The `main_loop` function is where Pygame does its heavy lifting. It's also where we inject the XLoBorg sensor readings into the ball's update method





# Aber SailBot

The autonomous Raspberry Pi-powered robot yacht built by British students that competes worldwide







## How did you get into making and racing autonomous sailing boats?

**Dan Clark:** It was because the opportunity arose, really. I mean, robotics has always been a great interest to me and then when the opportunity came up to do something like this, I jumped at the chance.

**Louis Taylor:** The department does quite a lot of research into autonomous sailing and they helped us start a team to actually build something for the [International Robotic Sailing Regatta (SailBot)] competition.

## Which competitions have you been in so far with the Aber SailBot?

**DC:** Last year in the summer we did SailBot 2013, which was in [Boston, Massachusetts, USA]. Then we did the World Robotic Sailing Championships, which was held in Brest in the north of France. This year we're going to be doing the same two competitions again, but in different places.

**LT:** This year SailBot is in [San Francisco, California, USA] in June and World Robotic Sailing Championship is in Galway, in Ireland.

## Are you particularly looking forward to one over the other?

**LT:** I think SailBot because we have quite a lot of... unfinished business there. We were narrowly beaten by the US Naval Academy by a few points [to first place].

## What made you choose the Raspberry Pi as a way to power your SailBot?

**LT:** We needed some sort of brains for the boat and it needed to be somewhat low-powered, somewhat compact enough to fit inside the boat. We wanted to



**Dan Clark** was an Aberystwyth University student at the time of writing, reading computer science and artificial intelligence



**Louis Taylor** was also at Aberystwyth University studying computer science and artificial intelligence. The two met during their first year tutorials



have the freedom to choose pretty much any language we wanted and obviously the Raspberry Pi lets us do that because it's running a full Linux operating system. It meant we could develop software on our Linux laptops, debug it on the laptops and then just put it on the Pi and it would work instantly. Essentially we wanted something that would run on Linux.

**DC:** We already know how to use it, so it saved us from having to learn anything that's particularly new.

### **Had you used the Raspberry Pi in any projects before the Aber SailBot?**

**LT:** I had. I've also used the BeagleBone and the BeagleBoard, which are both considerably more expensive than the Pi.

### **The Raspberry Pi Foundation is one of the sponsors for the SailBot, how did that come about?**

**DC:** In May last year we went to Cambridge to present the boat at the Cambridge Raspberry Pi Jam; as it turned out Jack Lang – one of the founding members of the Raspberry Pi foundation – was there. He saw us and our presentation where we said we needed some money and he came up to us at the end to tell us the Foundation could probably help us with that.

**LT:** It was pretty good, because at that point we were uncertain as whether we could actually go to the competition or not, because we hadn't raised nearly enough money to actually get ourselves to Boston. So with a month to go to the competition, we were still not sure whether we were actually going. We had a boat that worked, but no money.

### **If you like**

Interested in the robotics and automation part of this project? Then check out Dawn Robotics for Raspberry Pi robot kits:

**dawnrobotics.co.uk**

### **Further reading**

To learn more about the Aber Sailbot, visit their website at:

**abersailbot.co.uk**





**You're working on a second version of the SailBot, will it be ready for the competitions this year?**

**LT:** Yes. So at the moment, we've been inundated with coursework, exams and other university work over the last few months. But now term is finishing, we have a good three weeks to put some really solid work on the boat. So essentially we're going to work full-time on it over the next couple of weeks.

**DC:** It should be ready to sail, at least partially, in the next two weeks.

**LT:** At this point we can stop doing hardware and focus more on the electronics and soldering.

**Below** The Raspberry Pi is safely stowed inside the hull in order to prevent it from getting damaged by the water



## Once the second iteration of the SailBot is complete, what will be the fate of the original boat?

DC: That at the moment is currently kind of designated as a research boat to an extent. We've had a few ideas of scientific papers we'd like to write around the topic. So this boat will allow us to implement the experiments we need to do.

LT: Also it's good to have two platforms for our software, because the electronics and control system in terms of hardware is almost identical. We can pretty much put the same code on both boats and it should work. So we can use that as a test for the other boat.

## What else have you been using the Raspberry Pi for?

LT: I've used it for a small robot in the past with another robotics team, so that was a small cute little thing that picked up cubes and used vision to detect where it was. I'm also working on another similar robot, as my pet project.

## Do you have any other projects you want to do in the future that is Raspberry Pi related?

DC: For a long time we've wanted to do a quadcopter that's Pi-powered.

## Finally, what's your current state in terms of sponsorship?

LT: We're quite actively looking for sponsorship; our current main sponsors are QinetiQ, ARM and Kano computing. We've currently raised £2,500 this year, but still need £900. You can find out more information on how to sponsor us from the Aber Sailbot website.

“We're quite actively looking for sponsorship. You can find out more information on how to sponsor us from the Aber Sailbot website”





# Talking Pi

Join the conversation at...



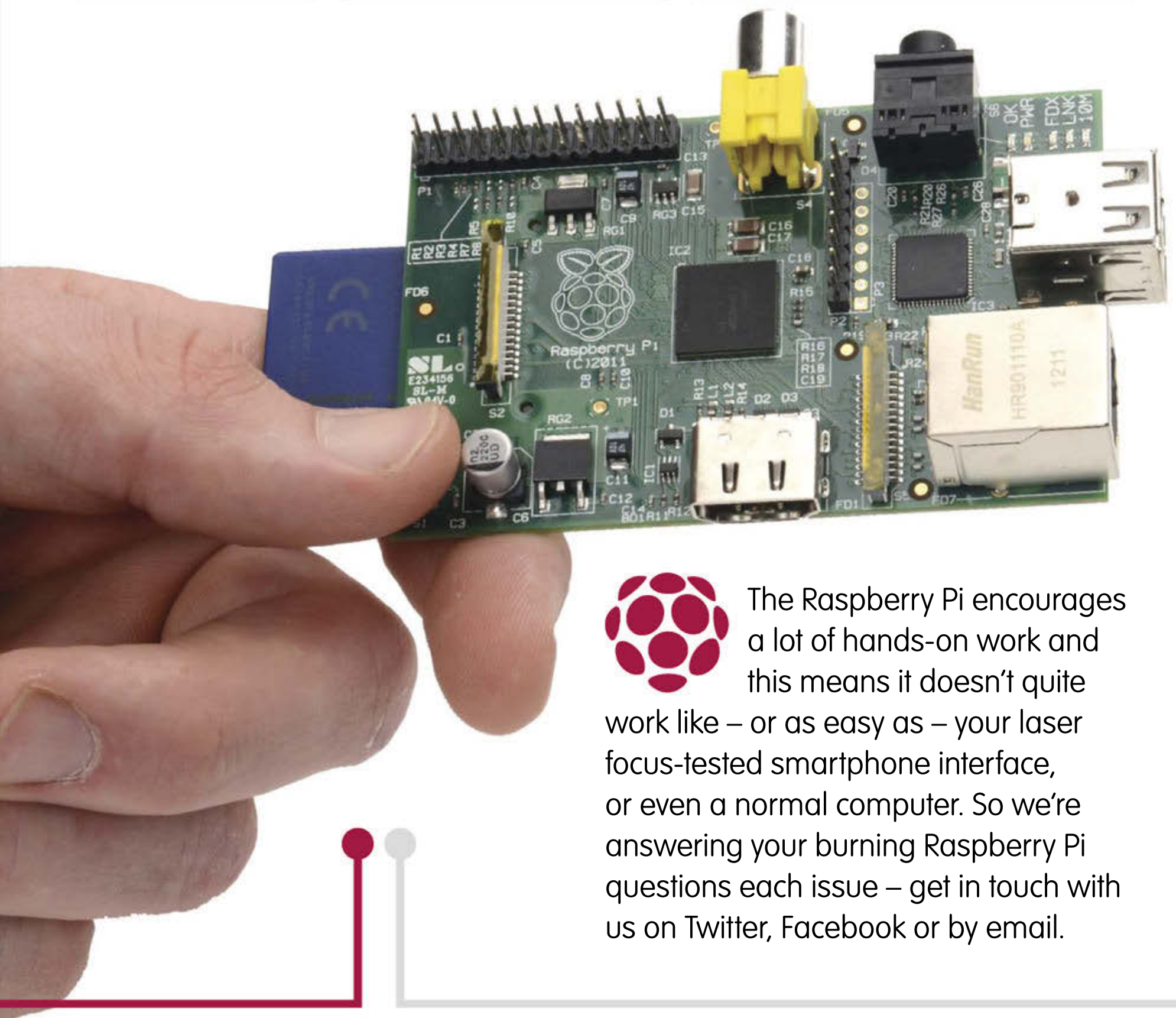
@linuxusermag



Linux User & Developer



RasPi@imagine-publishing.co.uk



The Raspberry Pi encourages a lot of hands-on work and this means it doesn't quite work like – or as easy as – your laser focus-tested smartphone interface, or even a normal computer. So we're answering your burning Raspberry Pi questions each issue – get in touch with us on Twitter, Facebook or by email.



Can I get an on/off switch for the Raspberry Pi?

Tracey via Facebook

Yes and no – there's no real traditional on and off switch for the Raspberry Pi that you can plug onto the board or whatever. You really have two options to do this: you can either wire a special one up to the GPIO port, or you can add a switch to the USB cable that powers the Pi.

The latter is generally a lot better and cheaper and can easily allow you to lengthen the USB cable that powers the Pi in the process. You can grab one from Pimoroni and a few other online retailers if your Google Fu is strong enough.



I like to use the video-out on my Raspberry Pi – can I use the B+ or A+ models?

Brent via email

Unfortunately, both of the newer + models of the Raspberry Pi lack the video-out in favour of just using an HDMI port. Adaptors don't work very well between them, however you can still buy and use the older model Raspberry Pi models, which do have these ports readily available. Generally speaking, use of the video-out port is probably very much a minority case, which is why it was removed from the + models in the first place.



Follow @LinuxUserMag on Twitter. Search for the hashtag #RasPiMag

## Win three great prizes



**PiKon telescope kit:** help write software for this 3D-printed, Pi-powered telescope to win one. [bit.ly/1FW8blN](http://bit.ly/1FW8blN)



**HDMI Pi screen:** win a 9-inch HD tablet-style screen by answering a simple question. [bit.ly/1wEdq4p](http://bit.ly/1wEdq4p)



**Mini speaker:** answer another easy question to win a smart wireless speaker for your Pi. [bit.ly/1u8ipwv](http://bit.ly/1u8ipwv)

Can I use  
an infrared  
remote on my  
Raspberry Pi?  
**Christos via  
email**

There are various methods that let you do this – most USB IR sensors will work just fine with XBMC/Kodi and are relatively cheap, but they do require a universal remote of some kind. Otherwise, you can try

out the Flirc – it's an IR receiver that can be programmed to accept specific signals as different inputs, allowing you to use an old or spare remote to control anything that allows for IR input.



Can I use my  
Raspberry Pi as  
a smartphone?  
**Rose via  
Facebook**

There have been some fun hacks with the Raspberry Pi to do a few things phone-wise with it. There's the straight-up Pi phone that was hacked together with a Raspberry Pi, a touch

screen and a special mobile phone radio chip to supply a signal to the device. It's all open source and readily available to use, but quite simple – the developer left it for others to tinker with. Otherwise, there were some Android ports in the early days of the Pi that allowed you to do some Android tasks with it.



## Win a FUZE for your Pi

We're giving away five FUZE's – for a chance to win one you just have to download FUZE BASIC onto your Pi, have some fun experimenting with it and then email us your most creative FUZE BASIC project (there are more details via the competition link below). The closing date for competition entries is **5 December 2014**, so send your project soon for a chance to win!



More info:  
[www.linuxuser.co.uk/  
news/win-a-fuze](http://www.linuxuser.co.uk/news/win-a-fuze)



# Next issue

 Get inspired  Expert advice  Easy-to-follow guides

## Program Minecraft

**Plus** Home media centre, reading robot and Bigtrak upgrade

Get this issue's source code at:  
[www.linuxuser.co.uk/raspicode](http://www.linuxuser.co.uk/raspicode)